자동화 시스템에서 Profibus 네트워크 인터페이스 구현 및 성능 평가

A Case Study on the Implementation and Performance Evaluation of Profibus Network in Automation Systems

김 기 암, 홍 승 호
(Ki Am Kim and Seung Ho Hong)

Abstract : This paper presents an implementation method of Profibus interface software using FMS(Fieldbus Message Specification). The Profibus interface software is implemented on PC which is widely used as an industrial computer as well as a commercial embedded controller called IUC(Intelligent Universal Controller). In order to enable the Profibus interface software to handle many application tasks and communication services, two kinds of real-time/multi-tasking operating system, OS-9 and CTask, are utilized. We also develop an experimental model of Profibus-based automation system, and evaluate the performance of the Profibus network. Through experiments, the user layer level message latency is measured with respect to the change of message length, message generation interval and TRT(Target Rotation Time). The results of experiment are compared with those of a simulation model which comprises only the physical and data link layers of Profibus. The results of this study shows that the message latency in the user layer level occupies fairly a large part of the total message latency.

Keywords : Profibus, FMS, interface S/W, implementation, automation system, experiment, performance

I. 서론

최근 수년간에 걸쳐 컴퓨터 및 통신 기술이 급속하게 발전하면서 통신망을 이용한 자동화 시스템의 보급이 크게 증가하고 있다[1][2]. 1980년대 초반부터 생산자동화 환경에서 이기종의 자동화 장비들 간의 통신을 위한 표준화된 통신망으로 MAP(Manufacturing Automation Protocol)이 개발되었다[3][4]. MAP은 공장 자동화 환경에서 매우 다양한 통신 기능을 제공하는 네트워크 시스템이지만 구조상 OSI (Open Systems Interconnection) reference model[5]에서 제시하는 7계층을 모두 가지고 있어 자동화 현장의 필드에 설치된 각종 자동화 기기들간의 실시간 통신을 지원하기에는 적합하지 않은 시스템으로 인식되고 있다. 1980년대 후반부터 자동화 현장에 설치된 각종 제어 및 자동화 관련 장비들에서 생성되는 데이터들의 실시간 통신을 지원하는 동시에 가격이 저렴한 네트워크 시스템의 필요성이 제기되었으며, 이러한 요구 사항을 만족시키기 위하여 개발된 네트워크 시스템이 필드버스이다[6][7].

필드버스가 출현하기 이전에 필드 장비들 간의 통신은 주로 4~20mA등의 아날로그 신호들이 사용되었다. 아날로그 신호 전송의 단점은 배선이 매우 복잡해지는 동시에 신호 전달 과정에서 외란에 민감하게 영향을 받는다는 것이다. 기존의 아날로그 신호를 이용한 통신 방법을 필드버스로 대체함으로써 얻을 수 있는 직접적인 효과로는 배선에 소요되는 비용의 절감과 전송 신호를 디지털화함으로써 데이터 전송의 신뢰도가 향상된다는 것들이 있다. 이러한 직접적인 효과 이외에 필드버스에서는 동일한 배선으로 여러 개의 센서들에서 발생하는 중복 신호를 동시에 처리할 수 있고, 양 방향 통신을 통하여 네트워크 상의 각종 기기들을 모니터링할 수 있을 뿐만 아니라 센서의 주기적 보정과 같은 조치를 네트워크를 통하여 자동으로 수행할 수 있어 시스템의 운용 및 유지 보수에 소요되는 비용을 절감할 수 있다. 이와 더불어 프로세서와 메모리가 내장된 스마트센서와 같은 성능이 향상된 필드 기기들의 도입이 가능하여짐에 따라 자동화 시스템의 성능이 향상될 수 있다. 또한, 필요한 경우에 시스템에 새로운 기능을 추가하거나 불필요한 기능을 삭제하는 등의 시스템 변형이 용이하여짐에 따라 시스템의 유연성과 확장성이 향상된다. 이러한 필드버스의 표준이 정착된다면 자동화 시스템의 사용자는 더 이상 폐쇄적인 제어 및 자동화 장비를 공급하는 생산 업체에 기술적으로 종속 당하지 않고 자체적으로 다양한 시스템을 구축할 수 있을 것이다.

필드버스에는 국제 표준안으로 규격이 작성 중에 있는 IEC/ISA 필드버스를 비롯하여 이미 제품화가 완료된 Profibus와 WorldFIP 및 최근에 개발이 완료된 Foundation Fieldbus 등이 있다. 그밖에도 이러한 필드버스들에 비하여 제한된 기능만을 제공하는 센서버스들이 상용화되어 사용되고 있으며, 이러한 센서버스에는 CAN, Interbus 등이 포함된다[8]. 현재 유럽, 미국, 일본 등의 기술 선진국에서는 필드버스를 이용한 자동화 시스템의 구축이 매우 활발히 진행되고 있다. 그러나, 국내에는 아직 필드버스를 이용한 자동화 시스템의 구축 기술이 축적되지 못한 상태에 있으며, 따라서 자동화 시스템에 필드버스를 도입한 사례가 매우 드물다. 즉, 각종 산업 자동화 시스템에 필드버스를 도입하려 해도 관련 지식 및 기술의 부족으로 필드버스 구축 방법에 대하여 알지 못하고 있는 상태이며, 이것이 국내의 자동화 시스템에 필드버스를 도입하는데 있어서 가장 큰 장애 요인으로 작용하고 있다. 자동화 시스템에 필드버스를 도입하기 위하여 가장 먼저 해결하여야 할 사항은 각종 자동화 장비를 필드버스의 통신 시스템에 접속시키기 위한 인터페이스 소프트웨어를 개발하는 것이다. 본 논문에서는 산업 설비에

접수일자 : 1997. 3. 5., 수정완료 : 1998. 1. 20.
김기암 : 현대정보기술(주) 산전정보기술연구소
홍승호 : 한양대학교 제어계측공학과
※ 이 연구는 96년도 한국과학재단 핵심전문연구과제 연구비지원에 의한 결과입니다.(과제번호: 961-0924-138-2)

서 자동화 장비들을 필드버스 네트워크에 접속시키기 위한 핵심 기술인 필드버스 접속 소프트웨어의 구현 방법을 제시한다. 본 논문에서 제시하는 필드버스 접속 소프트웨어의 구현은 Profibus의 응용계층인 FMS(Fieldbus Message Specification)를 기반으로 하여 개발된다. Profibus는 이미 제품화가 완료되었으며, 각종 아날로그 및 디지털 변수의 입출력 기능을 비롯하여 파일 전송, 프로그램의 원격 기동, 자동화 시스템에서 발생하는 각종 사건의 처리 기능들을 제공하여 각종 자동화 시스템에서 요구하는 대부분의 통신 기능을 만족시킬 수 있는 프로토콜이다.

자동화 장비의 응용 프로그램에서는 복잡하고 다양한 작업을 동시에 처리하여야 하는 경우가 많이 발생한다. 본 논문을 통하여 제시하는 Profibus 접속 소프트웨어는 여러 개의 응용 서비스 및 통신 서비스가 동시에 수행될 수 있는 실시간 다중 작업 프로그래밍 환경 하에서 구현된다. 본 연구에서는 현재 자동화 시스템의 여러 분야에서 산업용 컴퓨터로 널리 사용되고 있는 PC와 모듈 형태로 구성되어 독립된 제어 기능을 수행할 수 있는 전용제어기인 IUC(Intelligent Universal Controller)에서 동작될 수 있는 Profibus 접속 소프트웨어를 개발하였으며, PC와 IUC에서의 실시간 다중 작업 프로그래밍 개발 환경은 각각 CTask와 OS-9이 사용되었다.

산업 현장에 필드버스를 도입하는 경우에 가장 우선적으로 고려하여야 할 사항은 실시간 데이터를 주어진 시간 이내에 전송이 완료되도록 하는 것이다. 이러한 실시간성을 보장하기 위하여서는 응용 계층 및 사용자 계층의 데이터 지연 시간까지를 고려하여 네트워킹 시스템을 설계하여야 한다. 그러나 현재까지 발표된 대부분의 연구 결과는 네트워크 시스템의 성능 평가를 위하여 시뮬레이션 모델을 사용하고, 이러한 시뮬레이션 모델은 데이터링크 계층까지 만을 모델링하고 있다. 본 논문에서는 일반적으로 데이터링크 계층까지만 모델링되는 시뮬레이션 모델을 네트워크 시스템의 설계에 적용하는 경우에 대한 타당성을 검증하고, 이의 보완을 위하여 추가로 어떠한 사항들이 고려되어야 하는 가를 제시한다. 이를 위하여 본 논문에서는 Profibus 접속 소프트웨어를 이용하여 자동화시스템에 적용될 수 있는 실험 모델을 구축하고, 이러한 실험 모델을 통하여 Profibus 시스템의 성능 실험을 수행한다. Profibus 프로토콜에서 네트워크 시스템의 성능에 민감한 영향을 미치는 파라미터들로는 메시지 발생 주기, 메시지 길이 및 TRT(Target Rotation Time) 등이 있으며[9], 본 논문에서는 이러한 네트워크 파라미터 값들의 변화에 대하여 사용자계층까지를 포함하는 메시지 지연시간을 측정한다. 본 논문에서는 이러한 실험 결과를 Profibus의 데이터링크계층까지만이 모델링된 시뮬레이션 모델의 모의실험 결과와 비교하여, 사용자 계층까지를 포함하는 전체 메시지 지연시간과 데이터링크 계층까지의 데이터 지연시간과의 관계를 비교한다.

본 논문은 모두 다섯 장으로 구성된다. II장에서는 Profibus의 특성과 구조 및 Profibus의 응용계층인 FMS와 FMA7의 기능 및 구조에 대하여 간략히 기술하였다. III장에서는 본 연구를 통하여 구현된 네트워크 접속 소프트웨어의 구조와 기능에 대하여 기술하였으며, IV장에서는 Profibus 접속 소프트웨어가 구현된 자동화 시스템의 실험 모델을 이용하여 메시지 지연 시간을 측정하고, 측정된 실험 결과를 시뮬레이션 모델의 모의 실험 결과와 비교, 분석한다. 마지막으로 V장에서는 본 논문의 결론과 추후 연구 사항에 대하여 기술한다.

II. Profibus 특성 및 구조

본 장에서는 Profibus의 각 계층별 기능과 특성에 대하여 간략히 기술한다[10]. Profibus는 독일과 유럽에서 필드버스의 표준으로 제정한 시스템으로 이미 제품화가 완료되어 각종 자동화 시스템에서 널리 사용되기 시작하고 있다. Profibus의 계층 구조는 물리계층, 데이터링크계층 및 응용계층으로 구성되며[11][12], 사용자는 응용계층에서 제공되는 각종 서비스를 이용하여 사용자가 구축하려는 자동화 시스템의 환경에 적합한 사용자 계층의 Profibus 접속 프로그램을 구현하여야 한다.

물리계층은 전달될 데이터를 전송 신호로 변환하여 전송 매체를 통해 이를 전송하고, 수신된 전송 신호를 다시 데이터로 변환하는 계층이다. 버스 토폴러지를 사용하는 Profibus는 리피터를 사용하지 않는 경우에 하나의 세그먼트에 최대 32개의 노드가 접속될 수 있으며, 3개의 리피터를 사용하여 노드 수를 최대 127개까지 확장할 수 있다. Profibus 규격서에는 최대 데이터 전송 속도가 200m의 버스 길이에서 500Kbps로 주어져 있으나, 최근에는 1.5Mbps와 12Mbps의 전송 속도를 갖는 제품도 개발되었다. 네트워크 시스템 구성 시 버스 길이와 데이터 전송 속도는 매우 밀접한 관계를 가지고 있으므로 각별히 주의를 기울여야 한다.

Profibus의 데이터링크계층은 FDL(Fieldbus Data Link Layer)이라 하며, 매체접속제어(medium access control) 기능과 논리링크제어(logical link control) 기능을 제공한다. 매체접속제어 기능이란 여러 개의 노드들이 하나의 네트워크 미디엄을 공유하는데 있어서 데이터간의 충돌을 방지하기 위하여 각각의 노드에 데이터를 전송하는 권한을 부여하는 기능을 말한다. Profibus에서는 매체접속제어 방식으로 토큰-패싱 방식과 폴링 방식을 제공한다. 논리링크제어 기능은 통신을 하려는 노드들 간에 논리적 링크 또는 통신 관계를 설정 및 해제하는 기능을 비롯하여 통신 과정에서 발생할 수 있는 각종 오류를 처리함으로써 노드들 간에 신뢰할 만한 데이터 전송이 이루어지도록 하는 기능을 수행한다. Profibus의 통신 서비스는 크게 connectionless와 connection-oriented의 두 가지로 나뉘어진다. Connectionless 서비스는 통신 관계를 설정하지 않고 데이터를 전송하는 서비스로 broadcast 방법과 multicast 방법이 사용되며, SDN (Send Data with No Acknowledge) 서비스를 이용한다. Connection-oriented service는 통신관계를 설정한 후 데이터를 전송하는 방법이다. 통신관계 설정시 중요한 파라미터로는 노드의 주소와 SAP(Service Access Point)이 있다. 주소는 데이터가 전송된 노드와 전송 될 노드를 정의하고, SAP은 상위 계층에 서비스를 제공하는 지점이다. Connection-Oriented 방식에서 제공되는 서비스로는 SDA (Send Data with Acknowledge), SRD(Send and Request Data with Reply), CSRD(Cyclic Send and Request Data With Reply)가 있다.

Profibus 응용계층에는 FMS(Fieldbus Message Specification), LLI(Lower Layer Interface) 및 FMA7(Fieldbus Management Application)이 있다[12]. FMS는 통신 객체와 응용 서비스를 정의하며, LLI는 응용 서비스와 FDL를 접속하는 기능을 가지고 있다. FMA7은 네트워크 관리 서비스를 수행한다. 응용계층에서는 전송할 데이터인 PDU (Protocol Data Unit)를 생성한다. 즉 사용자가 응용계층에게 데이터 전송 서비스를 요청하면 응용 계층에서는 이를 네트워크를 통하여 전송하기 위해서 PDU라는 패킷을 만든다. 일반적으로 패킷은 원하고자 하는 서비스에 대한 정보

(서비스 종류 및 파라미터)를 포함하여야 한다. 이것은 Syntax라는 미리 정의된 구조를 갖게 되는데 Profibus에서는 ISO ASN.1(Abstract Syntax Notation One)을 따른다[13][14].

FMS는 Profibus의 응용계층 가운데 필드버스의 사용자와 연결되는 부분이다. FMS는 산업 현장에서 사용되는 자동화 기기의 응용 프로그램에 접속되며, 이러한 기기들은 상호 호환성 및 독립성 등이 보장되어야 한다. 그러므로 FMS를 구현하기 위해서는 객체 지향적인 방법이 사용된다. FMS는 VFD(Virtual Field Device)관리, OD(Object Directory)관리, 연결(context)관리, 변수(variable)관리, 사건(event)관리, 영역(domain)관리, 프로그램 기동(program in- vocation)관리, 접근 보안(access protection)관리 등의 기능을 제공한다. FMA7(Fieldbus Management Layer 7)에서는 Profibus 네트워크의 관리 역할을 수행한다. 즉 버스 시스템 전체의 구성과 유지 및 보수의 역할을 하며 이는 개방화된 통신을 보장하기 위해서 매우 중요하다. FMA7 서비스의 요청은 내부(local)와 원격(remote)에서 모두 가능하다. 원격 서비스는 통신 상대방에게 FMA7 서비스를 요청하는 것으로써 상대방은 자신의 내부 서비스를 이용하여 이를 실행하고 결과를 통보해 주어야 한다. FMA7에서는 연결 관리(context-management) 구성 관리(configuration-management) 및 오류 관리(fault-management) 등의 기능을 제공한다.

응용계층의 통신 서비스를 수행하기 위해서는 서비스를 요구하는 클라이언트와 클라이언트의 서비스 요구에 응답하는 서버와의 관계를 설정하여야 한다. Profibus의 응용계층에서 클라이언트와 서버 관계는 통신 객체, 통신 서비스 및 통신 관계 등의 통신 요소들로 구성된다. 통신 객체는 각 노드 사이에서 교환되는 데이터를 말하며, 예를 들어 센서에서 측정되는 데이터를 비롯하여 프로그램을 포함하는 데이터 파일 또는 통신 관련 파라미터 등이 여기에 포함된다. 한 노드는 여러 개의 통신 객체를 가질 수 있으며, 이러한 통신 객체의 목록을 OD(Object Dictionary)라 한다. OD는 index에 의해 각 객체가 구별된다. 통신 서비스는 이들 객체들에 대하여 해당 노드의 응용 서비스를 수행한다. 통신 관계는 각 노드 간 서비스의 논리적 연결 기능을 수행한다. Profibus에서는 서버-서버 연결을 할 수가 없으므로 각 연결에서는 최소한 하나 이상의 클라이언트가 필요하다. Profibus에서는 통신 관계의 목록을 CRL(Communication Relationship List)에서 관리한다. 이 목록은 CR(Communication Reference) 순으로 기입되는데 각 CR은 해당 노드의 응용 프로세서들 간의 데이터 교환을 위한 관계를 설정한다. 실제로 Profibus 네트워크를 설계하고 구성할 때 먼저 이들 통신 요소를 정의하고 각 통신 요소간의 관계를 설정해야 한다. 따라서 네트워크 시스템이 매우 복잡한 경우에 이러한 통신 요소를 구성하는 도구를 사용하여 네트워크 시스템의 구성을 단순히 처리할 수 있도록 할 필요가 있다.

III. Profibus 접속 소프트웨어 구현

현재까지 국내에서는 자동화 현장에 필드버스를 도입한 사례가 극히 드물고, 따라서 필드버스 시스템의 구축을 위한 하드웨어나 소프트웨어의 기술 지원이 매우 부족한 실정이다. 본 논문에서는 자동화 현장에서 로봇, PLC, NC 머신, 각종 제어기 및 컴퓨터 등에 탑재된 응용 프로그램들이 profibus를 통하여 데이터와 파일 등을 전송할 수 있도록 하는 Profibus 접속 소프트웨어를 구현하는 방법을 제시한다. 본 장에서는 여러 개의 응용 및 통신 서비스들이 동시에 수행될 수 있도록 하기 위하여 실시간 다중 작업 프로그래밍 환경하에서 Profibus 접속 소프트웨어를 구현하는 방법을 제시한다. 본 연구에서는 현재 자동화 시스템의 여러 분야에서 산업용 컴퓨터로 널리 사용되고 있는 PC와, 모듈 형태로 구성되어 독립된 제어 기능을 수행할 수 있는 전용 제어기인 IUC(Intelligent Universal Controller)에서 동작될 수 있는 Profibus 접속 소프트웨어를 개발하였다. PC와 IUC에서의 실시간 다중 작업 프로그래밍 개발 환경은 각각 CTask와 OS-9이 사용되었다. 본 장의 제 1 절에서는 CTask를 이용한 PC용 Profibus 접속 소프트웨어 구현 방법에 대하여 기술하고, 제 2 절에서는 OS-9를 이용하여 IUC에서 Profibus 접속 소프트웨어를 구현하는 방법에 대하여 기술한다.

1. PC에서 Profibus 접속 소프트웨어 구현

산업용 PC는 자동화 시스템에서 범용 제어기, 데이터 처리 및 저장 장치 또는 모니터링 시스템 등 각종 응용 분야에 광범위하게 사용되고 있다. 현재 기술 선진국에서는 PC를 이용한 개방형 제어기의 연구가 매우 활발히 진행되고 있다[15][16]. 본 절에서는 Profibus의 기능을 PC에서 구현하는 경우에 대하여 기술한다. PC를 Profibus에 접속하기 위해서는 마이크로제어기를 이용한 표준화된 보드를 사용하여야 한다. 본 논문에서는 V25+를 내장한 CP5412 보드를 호스트 컴퓨터인 PC에 장착하였다. V25+는 8088/86 계열과 호환이 되고 두 개의 UART (Universal Asynchronous Receiver Transmitter)를 제공하며, 내부 타이머를 제공하고 외부의 타이머를 확장할 수 있다. 또한, 20개의 어드레스라인을 제공하여 1MB의 메모리까지 사용할 수 있다. CP5412 보드는 16BitAT 커넥터를 통하여 PC와 연결되며, PC와의 데이터 교환을 위하여 512Kbyte의 동적 메모리와 64Kbyte의 Dualport RAM을 제공한다. Profibus와의 연결은 9-pin D-sub 커넥터를 사용한다. Profibus 제어기로는 SPC라는 ASIC을 사용하며 전송 속도는 최고 1.5Mbps까지 지원한다.

필드버스에 접속되는 센서, 제어기 및 엑츄에이터와 같은 기기들 간에는 정해진 시간 내에 데이터 교환이 이루어져야 한다. 이러한 네트워크의 실시간성을 보장하기 위하여서는 노드 내에서 응용 작업과 통신 작업이 동시에 수행되어야 한다. 여기서 응용 작업은 네트워크에 접속되는 기기들을 구동하는 프로그램이 될 수도 있고 사용자에게 정보를 제공하는 프로그램이 될 수도 있다. 통신 작업이란 Profibus의 응용 계층과 데이터 및 정보를 교환하는 부분이다. PC에서 일반적으로 사용하는 운영체계인 DOS만으로는 앞에서 언급하였던 실시간 다중 작업 환경을 만족할 수 없으므로 본 연구에서는 다중 작업 프로그램 기법이 가능한 CTask를 이용하였다[17]. CTask에서는 여러 개의 작업을 동시에 실행하기 위해서 각 작업마다 CPU의 시간을 분할하여 사용하며, 작업들의 동작 시기를 조정하기 위해서 큐를 사용한다. 즉, 각 작업들은 미리 지정된 우선 순위를 가지며, 대기하는 작업들은 우선 순위대로 큐에 저장된다. 스케듈러는 현재 수행하는 작업을 스택에 저장하고 큐에서 가장 우선 순위가 높은 작업부터 처리한다. CTask에서는 작업간의 데이터 교환을 위하여 자원(resource), 플래그(flag), 메일박스(mailbox) 및 파이프(pipe) 등을 지원한다.

Profibus의 실제 사용자에게는 네트워크의 하부 구조 등은 중요하지 않고 단지 통신 상대와의 데이터 교환만이 중요한 의미를 갖는다. 그러므로 Profibus 접속 소프트웨어인 사용자계층의 응용 프로그램은 하위 계층의 통신 프로토콜들과의 데이터 교환만을 고려해 주면 된다. 즉 사용자 계층의 응용 프로그램은 응용 계층인 FMS와 FMA7 서비스를

이용하여 통신 상대에게 데이터를 전송하거나 수신할 수 있도록 구현하여야 한다. 본 연구에서는 Softing에서 제공하는 FMS와 FMA7[18]을 이용하여 응용 프로그램을 구현하였다. 그림 1에는 사용자 계층의 응용 프로그램과 하위 계층인 FMS/FMA7 및 LLI와의 데이터 교환에 대한 구조가 나타나 있으며, 이들 사이에는 Description 블록과 Data 블록이 있다. Description 블록은 현재 수행되는 서비스를 설명해주는 부분이다. 여기서 *comm_ref*는 CR(Communication Reference)로서 CRL에 미리 정의된 통신관계(즉, 통신 상대의 주소와 SAP 등)를 참조한다. *layer*는 서비스가 수행되는 계층으로 FMS와 FMA7을 구분해 주며, *service*는 FMS와 FMA7에서 제공하는 서비스를 말한다. *primitive*에는 request, indication, response, confirmation이 있으며, 클라이언트와 서버의 데이터 교환 절차는 이러한 서비스 primitive를 통하여 이루어진다. *invoke_id*는 각 작업의 ID이며, result는 서비스 실행 결과를 알려주는 것으로 성공(positive)과 실패(negative)가 있다. Data 블록은 실제 데이터가 기입되는 부분으로, *index*와 *obj_code*를 통하여 미리 정의된 OD를 참조한다. LLI에서는 Description 블록과 Data 블록을 이용하여 전송할 프레임을 만들어 이를 하위 계층인 FDL에 넘겨준다.

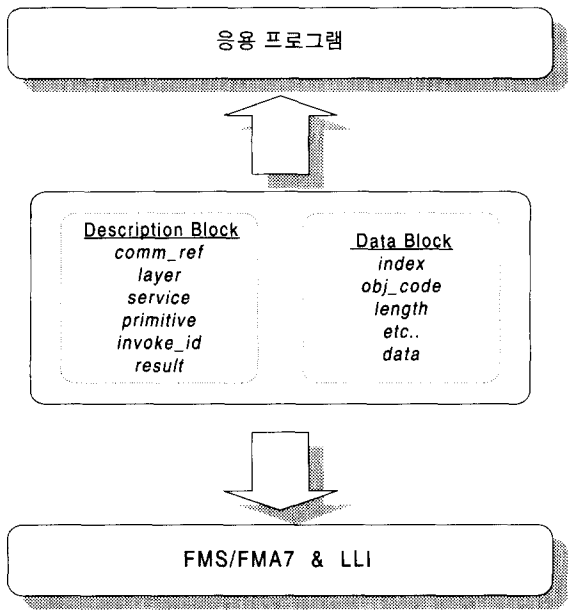


그림 1. FMS/FMA7과 응용 프로그램의 데이터 교환.
Fig. 1. Data exchange between FMS/FMA7 and application program.

그림 2는 FMS의 여러 기능들 가운데 변수 관리 기능의 Write서비스를 구현한 경우의 예에 대하여 실제 데이터가 교환되는 모습을 보여준다. 다른 서비스들과 마찬가지로 Write 서비스도 primitive에 의하여 데이터를 전송하거나 수신한다. 데이터를 전송하는 경우에는 다음과 같은 profi_snd_ req_res( ) 함수를 사용하여 전송하는 데이터의 Description 블록과 Data 블록을 참조할 수 있도록 한다.

```
int profi_snd_req_res
{
   PROFI_SERVICE_DESCR FAR  *sdb_ptr;
                            // Description 블록에 대한 포인터
   void     *data_ptr;      // Data 블록에 대한 포인터
}
```

Write.req 서비스의 Data 블록 구조는 다음과 같으며, 통신 상대에 전송할 데이터와 길이를 파라미터로 갖는다.

```
typedef struct VAR_WRITE_REQ
{
   T_ACC_SPEC     acc_spec;     // 접근 관리
   char       subindex;         // OD가 참조하는 부인덱스
   unsigned char  length;       // 데이터의 길이
   unsigned char  value[length]; // 실제 데이터
}
```

데이터를 수신하는 경우에는 다음과 같은 profi_rcv_ con_ind() 함수를 사용하며, 버퍼에 수신된 데이터를 참조하여 Description 블록과 Data 블록을 인가한다.

```
int profi_rcv_con_ind
{
   PROFI_SERVICE_DESCR FAR  *sdb_ptr;
                            // Description 블록에 대한 포인터
   void     *data_ptr;      // Data 블록에 대한 포인터
   unsigned int   *data_len; // 수신 데이터길이
}
```

Write 서비스 이외의 FMS에서 제공하는 다른 서비스들도 이와 유사한 방식으로 구현되며, 자세한 사항은 생략하기로 한다.

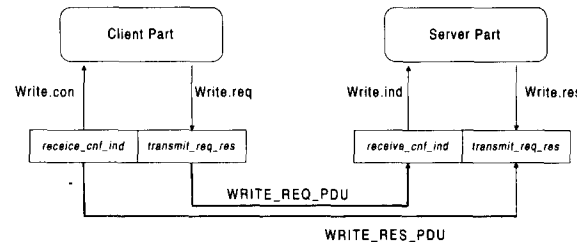


그림 2. Write 서비스의 데이터 교환.
Fig. 2. Data exchange in the write service.

그림 3은 CTask를 이용하여 개발한 네트워크 접속 프로그램의 전체 구조이다. Profibus 접속 소프트웨어에서 동시에 수행되어야하는 작업으로는 *receive_cnf_ind, transmit_req_res* 및 FMS가 있고 응용 프로그램은 필요한 만큼 추가할 수 있다. 노드에 새로운 데이터가 수신되면 *receive_cnf_ind* 태스크는 수신된 데이터에 인덱스 번호를 부여하고 수신데이터에 대한 Description 블록과 Data 블록을 저장한다. 인덱스 번호는 수신된 데이터의 식별자로 사용되며, 수신된 데이터의 인덱스 번호는 index_queue에 저장한다. FMS는 index_queue를 통하여 새로운 데이터가 수신되었는가를 확인하며, FMS가 작업 도중 새로운 인덱스를 발견하는 경우에 해당되는 Description 블록을 참조하여 인덱스 번호를 데이터가 수신되어야 할 응용 프로그램의 application_index_queue로 전달한다. 응용 프로그램 역시 application_index_queue를 통하여 새로운 데이터가 수신되었는가를 확인하며, 응용 프로그램이 작업 도중 새로운 인덱스를 발견하는 경우에 인덱스 번호에 해당되는 Data 블록에 저장된 데이터를 수신하고, 수행 중이던 작업을 계속한다.

응용 프로그램이 작업 도중에 데이터를 전송하여야 할 경우가 발생하면 응용 프로그램은 전송되어야 할 데이터의 Description 블록과 Data 블록을 생성하고, 메일박스 기능을

이용하여 이를 *transmit_req_res*에 전달한다. *transmit_req_res*가 메일박스에서 새로이 전송되어야 할 데이터를 탐지하면, *transmit_req_res*는 해당 데이터의 Description 블록과 Data 블록을 통하여 PDU(Protocol Data Unit)을 생성하고, 이를 하위 계층인 FDL(Fieldbus Data Link)로 전달한다. 본 논문을 통하여 개발된 Profibus 접속 소프트웨어는 실시간 다중 작업 프로그래밍 환경에서 구현되어 응용 프로그램이 자신의 작업을 수행하는 도중에도 수시로 데이터를 전송하거나 수신할 수 있다.

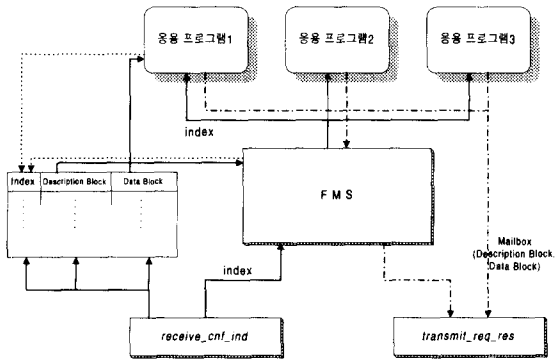


그림 3. PC 상에서 개발한 네트워크 접속 프로그램의 구조.
Fig. 3. Structure of PC-based network interface program.

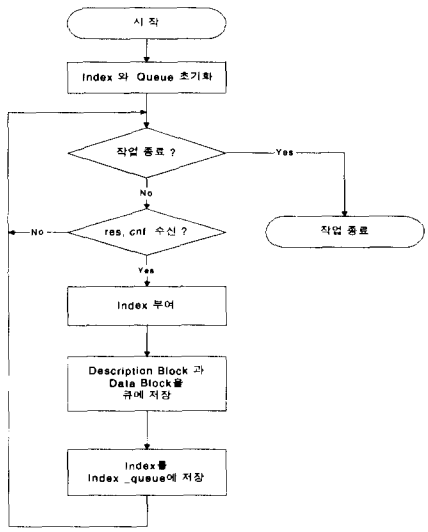


그림 4. *receive_cnf_ind* 작업의 순서도.
Fig. 4. Task sequence of *receive_cnf_ind*.

그림 4는 *receive_cnf_ind* 작업의 순서도이다. *receive_cnf_ind*는 하위 계층으로부터 데이터를 수신하는 역할을 하며, 작업이 시작되면 먼저 인덱스와 index_queue를 초기화한 다음, 수신 버퍼를 확인한다. *receive_cnf_ind*가 FDL 계층으로부터 confirm 또는 indicate primitive를 통하여 새로운 데이터가 수신되었음을 확인하면, 해당 데이터의 Description 블록과 Data 블록을 큐에 저장하고 수신데이터에 인덱스 번호를 부여한다. 그리고 인덱스 번호를 index_queue에 전달하여 FMS가 이를 확인하도록 한다.

그림 5는 *transmit_req_res* 작업의 순서도이다. *transmit_req_res*는 메일박스를 검사하여 전송할 데이터가 있는지 검사한다. 전송할 데이터가 있는 경우에는 Description 블록을 참조하여 수행해야 할 서비스(즉, 서비스 종류 또는 서비스 프리미티브 등)를 선택한다. 또한 Data 블록을 통하여 전송할 데이터 프레임을 생성한 후 FMS나 FMA7 서비스를 통하여 하위 계층인 FDL에 데이터 전송을 요구한다.

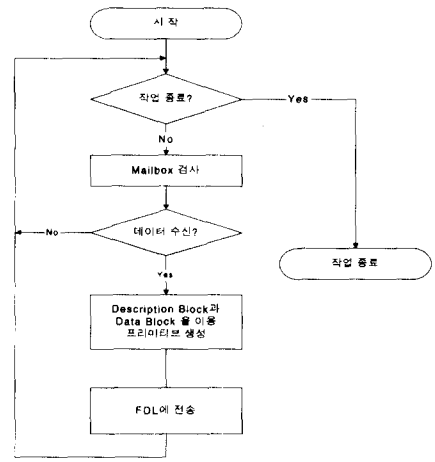


그림 5. *transmit_req_res* 작업의 순서도.
Fig. 5. Task sequence of *transmit_req_res*.

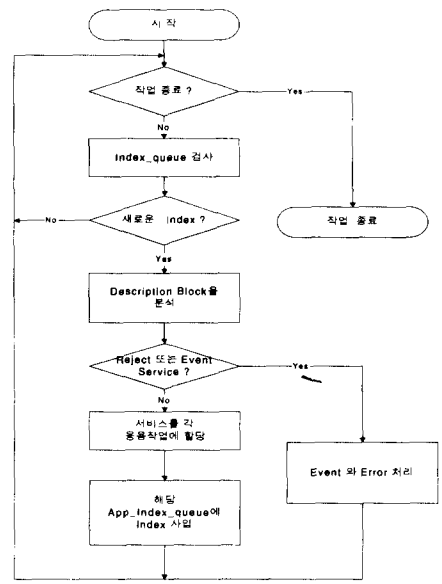


그림 6. FMS 작업의 순서도.
Fig. 6. Task sequence of FMS .

그림 6은 FMS 작업의 순서도이다. FMS는 index_queue에 새로운 인덱스가 발생되었는가를 검사함으로써 새로이 수신된 데이터를 탐지한다. 만일 새로운 인덱스가 발견되면, 수신된 데이터의 Description 블록에 기록된 서비스 종류와 프리미티브 등을 분석하여 수행되어야 할 서비스를 선택한다. 이때 수신된 데이터가 사건 처리나 오류가 발생에 관련된 경우에는 이를 처리하고, 그렇지 않은 경우에는 인덱스를 해당 application_index_queue에 할당함으로써 수신 데이터를 해당 응용 프로그램에 인가한다.

그림 7은 응용 프로그램의 한 예를 보여준다. 응용 프로그램에서는 application_index_queue를 검사하여 새로운 데이터가 수신됐는지를 검사한다. 응용 프로그램이 작업을 수행하는 도중에 새로운 인덱스가 탐지되면 이에 대한 Data

블록을 참조하여 응용 프로그램의 해당 데이터를 갱신하고 다시 수행 중이던 응용 프로그램의 작업을 계속한다. 응용 프로그램에서 전송할 데이터가 발생하면 해당 데이터의 Description 블록과 Data 블록을 생성하고, 메일박스를 이용하여 이를 *transmit_req_res*에 전달한다.

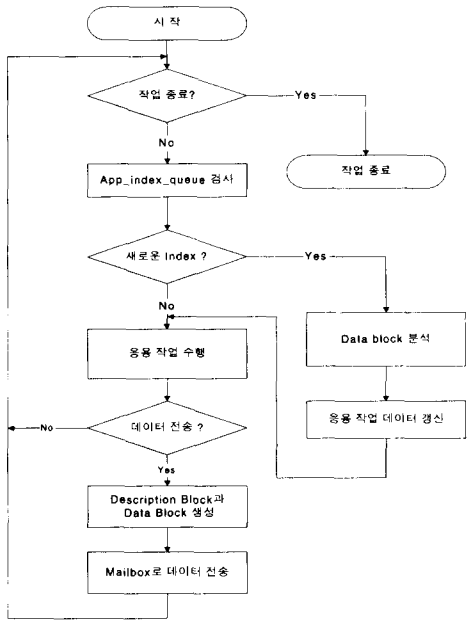


그림 7. 응용 프로그램의 순서도.
Fig. 7. Sequence of application program.

2. IUC에서 Profibus 접속 소프트웨어 구현

자동화 시스템에서는 특정한 제어 또는 자동화 기능이 내장된 embedded 방식의 전용제어기를 사용하는 경우가 많다. 본 장에서는 PEP사에서 모듈 형태의 전용제어기로 개발된 IUC에서 OS-9 오퍼레이팅 시스템을 이용하여 개발한 Profibus 접속 프로그램의 구조에 대하여 기술한다. IUC는 MC68302 IMP(Integrated Multiprotocol Processor)로 구성된 모듈형태의 제어기이다. 즉, 확장 모듈인 CXC를 사용하여 I/O와 같은 보드들을 추가로 확장할 수 있다. IUC는 DMA를 가진 세 개의 직렬 포트와 세 개의 타이머, 인터럽트 제어기, 범용 DMA 채널, 그리고 1152바이트의 dual-port RAM을 제공하며, 내장 프로그램이 저장될 수 있는 SRAM과 EPROM이 제공된다[19]. MC68302 프로세서는 68000 core와 SIB(System Integration Block) 및 CP(Communication Processor)의 세 개의 부분으로 구성된다[20]. SIB는 주변기기들과의 접속시 필요한 어드레스 디코딩, DTACK 발생, 버스 모니터링 등을 단순화하여 효율적인 하드웨어의 설계를 가능하게 한다. SIB에는 인터럽트 제어기와 세 개의 시스템 타이머가 제공되고 메모리 및 I/O와의 데이터 전송을 담당하는 IDMA(Independent DMA)가 있다. DPRAM(Dual Port RAM)은 CPU, CP, IDMA 및 외부 버스 마스터가 사용할 수 있다. OS-9은 680x0 계열의 마이크로 제어기를 구동하는 실시간 OS이다. OS-9은 모듈로 구성되며 시스템의 필요한 기능과 구성에 따라 추가하거나 삭제할 수 있다. 또한 시스템 관리자로서 실시간 커널을 제공하는데, 실시간 커널은 메모리, 프로세스 및 I/O 등을 관리하며 시분할에 의한 다중작업을 수행한다. 또한, 통합된 I/O의 관리를 위해 파일 관리자 및 디바이스 드라이버를 제공한다. 사용자와의 접속은 UNIX에서와 같이 쉘이 담당한다[21][22].

IUC에서는 실제 프레임의 전송이 CP를 통하여 이루어지므로 응용프로그램은 DPRAM을 이용하여 교환되는 데이터를 처리해야한다. 이는 PC의 경우에서와 동일하다. 따라서 프로그램의 기본 구조는 PC의 경우와 유사하다. 그림 8은 전체 프로그램의 구조이다. PC의 경우처럼 기본 작업으로 *receive_cnf_ind*와 *transmit_req_res* 및 FMS가 있고 응용 프로그램은 추가로 확장할 수 있다. 각 작업간의 데이터 전송은 파이프를 통하여 이루어지며 오류나 문제 발생 시 시그널과 사건을 이용하여 처리한다. *receive_cnf_ind, transmit_req_res*, FMS 및 응용 프로그램의 작업 순서도는 앞 절에 기술된 PC의 경우와 동일하므로 생략한다.

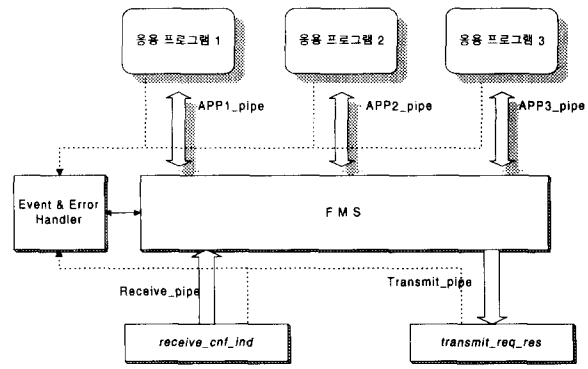


그림 8. OS-9을 이용한 네트워크 접속 소프트웨어 구조.
Fig. 8. Structure of network interface software using OS-9.

IV. 메시지 지연 시간 측정

필드버스에서는 데이터를 생성하는 여러 개의 노드들이 하나의 네트워크 미디엄을 공유하기 때문에 각각의 노드에서 생성되는 데이터를 적절히 관리 및 제어하지 못하는 경우에는 각각의 데이터들에 대한 응답시간 또는 처리량 등의 요구사항을 만족시키지 못하게 되어 자동화 시스템의 성능을 저하시킬 수 있다[23][24]. 메시지 지연 시간은 네트워크 시스템의 성능을 평가할 수 있는 가장 중요한 지표로서 사용자의 서비스 요구에 네트워크 시스템이 얼마나 빨리 응답할 수 있는 가를 나타낸다. Profibus에서 메시지의 지연시간에 영향을 미칠 수 있는 주요 요소들로는 접속되는 노드의 개수를 비롯하여, 메시지 발생주기, 메시지 길이 및 Target Rotation Time(TRT) 등이 있다[9]. Profibus에서 토큰이 논리링(logical ring)을 한 번 순환하는데 걸리는 시간을 Real Rotation Time이라고 한다. Profibus는 토큰을 받을 때 Real Rotation Time과 Profibus의 버스 파라미터로 지정되는 TRT를 비교하여 Real Rotation Time이 더 크면 우선 순위가 높은 데이터만을 전송하게 된다. 이는 각 노드에서 우선 순위가 낮은 메시지의 전송 시간을 제한함으로써 우선 순위가 높은 메시지에게 더 많은 전송 기회를 부여하기 위해서이다.

네트워크 시스템의 성능을 평가하는 방법에는 크게 해석적인 방법과 시뮬레이션 모델을 이용한 모의 실험을 통하여 네트워크 시스템의 성능을 간접 측정하는 방법 및 실제 실험 모델을 구축하여 네트워크의 성능을 직접 측정하는 방법 등의 세 가지가 있다. 해석적인 방법은 큐잉 이론을 통하여 네트워크 시스템의 수학적인 모델을 개발하고, 이를 통하여 네트워크의 성능을 평가하는 방법이다. 그러나 수학적인 모델을 개발하기 위하여서는 여러 가지 가정을 필요로 하며, 이러한 가정들에 대한 제약적인 조건으로 인하여 정확한 성

능 해석보다는 네트워크 시스템의 성능을 근사적으로 해석하는데 주로 사용된다[9]. 시뮬레이션 모델은 네트워크 시스템이 동작되는 과정을 컴퓨터를 통하여 그대로 모사 함으로써 해석적인 방법과 달리 가정을 필요로 하지 않으며, 따라서 해석적인 방법보다는 정확하게 네트워크 시스템의 성능을 평가할 수 있다. 시뮬레이션 모델은 그러나 주로 데이터 링크계층까지 만이 모델링된다. 왜냐하면 응용계층의 성능은 실제로 구현되는 자동화 장비의 하드웨어나 소프트웨어의 성능에 따라 크게 달라질 수 있기 때문이다.

실제의 자동화 설비에서 네트워킹 시스템은 그러나 데이터 링크 계층을 포함한 응용 및 사용자 계층까지의 데이터 지연 시간을 고려하여 설계되어야 한다. 사용자 계층까지의 데이터 지연 시간은 시스템의 하드웨어와 사용자 계층의 소프트웨어를 어떻게 구성하는가에 따라 달라지므로 실험 모델을 통하여 직접 측정하는 방법밖에 없다. 본 논문에서는 III장에서 제시한 Profibus 접속 소프트웨어를 이용하여 실험모델을 구축하고, 실험모델에서 측정된 사용자 계층까지의 데이터 지연 시간과 시뮬레이션 모델을 이용하여 얻어진 데이터 링크 계층까지의 지연 시간을 비교 및 분석한다. 이를 통하여 일반적으로 데이터 링크 계층까지만 모델링되는 시뮬레이션 모델을 네트워크 시스템 설계에 적용하는 경우에 있어서 타당성을 검증하고, 이의 보완을 위하여 추가로 어떠한 사항들이 고려되어야 하는 가를 제시한다. 본 장의 제1절에는 본 연구를 통하여 개발된 실험 모델이 기술되고, 제2절에는 시뮬레이션 모델의 구조가 기술되며, 제3절에서는 두 모델을 통하여 측정된 결과를 비교 분석한다.

1. 실험 모델 구성

본 장에서는III 장에서 기술한 Profibus 접속 소프트웨어를 이용하여 실제 자동화 시스템에 적용될 수 있는 실험 모델을 구축하고, 이를 통하여 네트워크를 통한 메시지의 지연시간에 대한 실험적 데이터를 도출함으로써 Profibus의 성능을 분석하고자 한다. 본 논문에서 구현한 자동화 실험 모델은 CP5412가 장착된 PC와 3대의 IUC 및 1대의 Smart I/O를 사용하였다. Smart I/O는 디지털 I/O 모듈과 PLC 기능이 부과된 IUC의 일종이다. 예를 들어, 생산 자동화 시스템을 구현하는 경우에 로봇 제어기, CNC 제어기 및 PLC 등과 같은 장비들은 PC, IUC 및 Smart I/O를 통하여 Profibus에 접속될 수 있다. 실험 모델에서는 또한 네트워크에서 발생하는 프레임의 상태를 측정하기 위해 버스모니터를 접속하였다. 이는 버스에 부하를 가하지 않고 단지 네트워크를 통하여 전송되는 프레임의 발생 시각을 측정하고 이를 분석하는 장비이다. 그림 9에는 자동화 시스템 실험 모델의 구성도가 나타나 있다.

실험 모델에서 메시지의 지연시간은 사용자 계층을 기준으로 측정되었다. 즉, 그림 10에서 보는 바와 같이 CP5412 보드를 장착한 PC의 응용 프로그램에서 IUC측에 Write 서비스를 요구하기 시작한 시점부터 이에 대한 응답이 오기까지 걸리는 시간을 측정한다. 이때 다른 노드들은 네트워크에 일정한 부하를 가한다. 실험에서 네트워크 시스템 관련 파라미터들의 설정은 표 1에 주어진 네트워크 파라미터들을 조합하여 여러 가지의 경우에 대하여 수행하였다. 표 1에서 bit time이란 하나의 bit을 전송하는데 소요되는 시간으로, 본 실험에서 데이터 전송 속도는 500Kbps이므로 1bit time은 2 μsec 에 해당한다. 본 실험에서는 실험 결과의 분석을 단순화하기 위하여 주어진 실험 조건에서 모든 노드의 메시지 발생주기, 메시지 길이, TRT는 동일한 값을 갖도록 하였다.

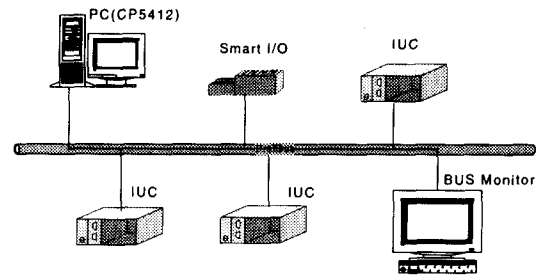


그림 9. 자동화 시스템 실험 모델.
Fig. 9. Experimental model of automation system.

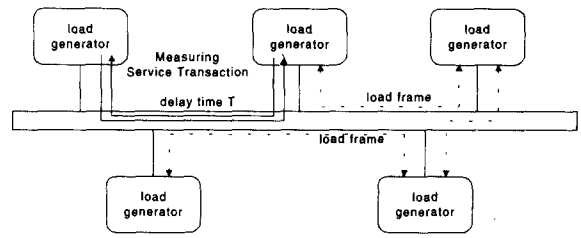


그림 10. 메시지 지연 시간의 측정.
Fig. 10. Measurement scheme of message delay time.

2. 시뮬레이션 모델 구조

Profibus와 같은 네트워크 시스템은 사건(event)이 발생할 때마다 시스템의 상태가 변화하는 이산 사건 시스템(discrete event system)으로 분류된다. 본 논문에서는 이산 사건 시스템의 모델링을 위한 전용 도구인 SIMAN/ARENA[25]를 이용하여 Profibus의 시뮬레이션 모델을 개발하였다. 이산 사건 시스템 모델링 기법의 기본 개념은 시스템의 상태가 시간의 변화에 대하여 어떻게 변화하는 가를 추적하는 기법으로, 시스템의 현 상태에서 앞으로 발생 가능한 모든 사건들 탐색하여 이들을 발생 시간의 순서에 따라 사건 스케듈링 리스트에 차례로 기록한다. 시뮬레이션 모델의 진행은 사건 스케듈링 리스트에 기록된 사건들을 발생되어야 할 사건의 순서에 따라 차례로 발생시킴으로써 시스템의 상태를 변화시키고, 변화된 상태에서 새로이 발생할 수 있는 사건들을 사건 스케듈링 리스트에 추가하는 방식으로 진행된다[25].

본 연구를 통하여 개발된 Profibus 시뮬레이션 모델은 크게 메시지 발생 부분과 메시지 전송에 관련된 프로토콜 부분으로 구성된다. 본 연구를 통하여 개발된 시뮬레이션 모델에서 사건을 스케듈링하는 절차에 대한 구조도가 그림 11에 나타나 있다. 먼저 시뮬레이션 초기화(*initialization*) 과정에서는 표 1에서 주어진 네트워크 관련 파라미터들이 입력된다. 초기화 과정에서는 네트워크 내의 모든 노드들에 대하여 *message generation* 사건을 스케듈링하고, 토큰의 발생을 위하여 *first token passing* 사건을 스케듈링한다. *message generation* 사건에서는 메시지를 발생시켜 메시지 관련 속성(전송 노드, 수신 노드, 메시지 길이, 발생 시간, 우선 순위 등)을 부여한 후 전송큐에 삽입한다. 또한 주어진 메시지 발생 주기에 따라 다음에 발생될 메시지를 스케듈링하여 이를 사건 스케듈링 리스트에 기록한다.

first token passing 사건에서는 *receive token* 사건을 스케듈링하여 토큰 전달 시간 후에 *receive token* 사건이 발생되도록 한다. *receive token* 사건에서는 각 노드에서 토큰을 수신한 후 Profibus 프로토콜 규격에 나타난 것과 동일한 과정을 거쳐 전송큐에 대기하는 메시지를 전달하는 과

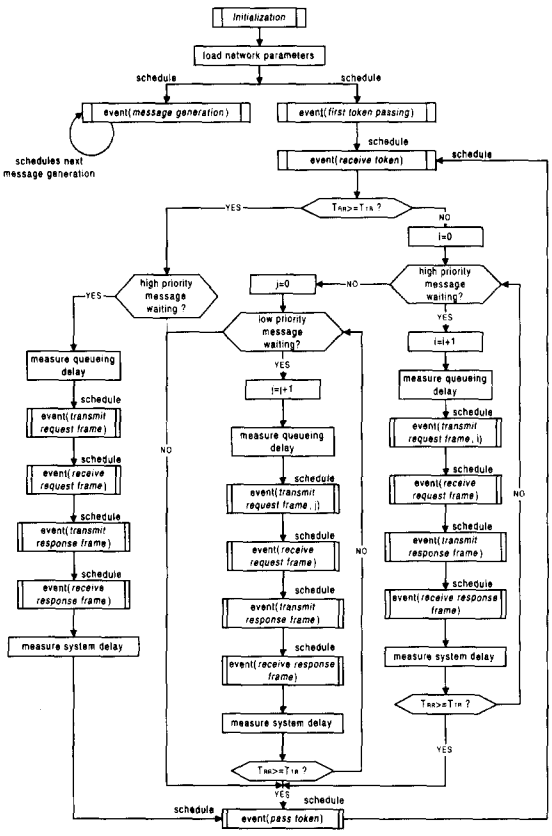


그림 11. 시뮬레이션 모델의 사건 스케듈링 구조.
Fig. 11. Structure of event scheduling in the simulation model.

정이 모델링된다. 즉, 토큰을 수신한 노드는 먼저 T_{RR}(Real Rotation Time)과 T_{TR}(Target Rotation Time)을 비교한다. 만일 T_{RR}이 T_{TR}보다 큰 경우에는 높은 우선 순위 메시지 하나만이 전송될 수 있다. 전송큐에 높은 우선 순위 메시지가 있는 가를 검사한 후 대기하고 있는 메시지가 없는 경우에는 다음 노드로 토큰을 전달하기 위하여 *pass token* 사건을 스케듈링한다. 그러나 메시지가 대기하고 있는 경우에는 토큰 처리 시간 후에 대기하는 메시지가 전송되도록 하기 위하여 *transmit request frame* 사건을 스케듈링한다. 이때 전송되는 메시지의 큐잉 지연시간을 계산한다. 송신 노드에서 *transmit request frame* 사건은 프레임 전송 지연 시간 후에 수신 노드에서 데이터가 수신되도록 *receive request frame* 사건을 스케듈링한다. 시뮬레이션 모델은 데이터 링크 계층의 지연 시간만을 고려하므로 수신 노드에서는 프레임을 수신한 즉시 *transmit response frame* 사건이 발생되도록 하며, *transmit response frame* 사건은 프레임 전송 지연 시간 후에 응답 프레임이 수신되도록 *receive response frame* 사건을 스케듈링한다. 응답 프레임이 도착하면 데이터 링크 계층에 데이터 프레임이 생성된 시점부터 이에 대한 응답 프레임이 도착한 시간까지의 지연 시간이 계산된다. 메시지 전송 노드는 응답 프레임이 도착한 후에 토큰을 다음 노드로 전송하기 위하여 다시 *pass token* 사건을 스케듈링한다. 만일 T_{RR}이 T_{TR}보다 작은 경우에는 T_{TR}이 만료될 때까지 여러 개의 데이터를 전송할 수 있으며, 그림 11에서 보는 바와 같이 높은 우선 순위 메시지를 우선적으로 전송한다. 메시지를 전송한 후에 토큰을 다음 노드로 전달하는 과정은 앞서 기술한 바와 동일하다. 시뮬레이션 모델에서 네트워크 시스템 관련 파라미터들의 설정은 표 1에 나타난 바와 같이 실험 모델의 경우와 동일하다.

표 1. 실험 모델 및 시뮬레이션 모델에서 네트워크 파라미터.
Table 1. Network parameters in the experimental and simulation models.

Target Rotation Time (bit-time)	5,000 / 7,500 / 10,000 / 12,500 / 15,000
Message Length (byte)	2 / 50 / 100 / 200
Message Generation Period (msec)	250 / 500 / 1,000
Slot Time (bit-time)	3,000
Baud Rate (Kbps)	500

3. 실험 결과와 시뮬레이션 결과의 비교

Profibus에서 메시지의 지연시간에 영향을 미치는 요소들로는 접속되는 노드의 개수를 비롯하여, 메시지 발생주기, 메시지 길이 및 TRT 등이 있다. 본 논문에서는 메시지 발생주기, 메시지 길이 및 TRT를 변화시켰을 경우에 데이터 지연 시간의 평균값이 사용자 계층까지를 포함하는 실험 모델과 데이터 링크 계층까지만을 모델링한 시뮬레이션 모델에서 어떻게 변화하는가를 비교, 분석하였다. 실험과 시뮬레이션은 표 1에서 주어진 다양한 조건하에서 수행되었으나, 실험 결과는 대부분 동일한 특성을 나타내었다. 따라서 본 논문에서는 메시지 발생주기, 메시지 길이 및 TRT를 변화에 대한 대표적인 결과만을 제시하기로 한다.

메시지 발생 주기의 변화

그림 12에는 네트워크 내의 모든 노드에서 200 바이트 길이의 메시지를 생성하고, TRT 값이 15000 bit time으로 설정된 경우에 발생 주기가 메시지 지연시간에 미치는 효과에 대하여 실제 측정 결과와 시뮬레이션 결과를 비교하여 보여준다. 그림 12를 보면 데이터링크 계층까지만 모델링된 시뮬레이션 결과에서는 메시지 발생주기의 변화가 메시지 지연시간에 미치는 영향이 크지 않은 것으로 나타났다. 이는 본 실험이 네트워크의 부하가 적은 상태에서 수행되었기 때문이다. 그러나, 트래픽이 낮은 경우에도 메시지 발생주기의 변화가 사용자 계층의 메시지 지연시간에 미치는 영향은 민감한 것으로 나타났다. 이는 메시지가 빈번하게 발생할수록 응용계층에서 메시지 처리에 대한 부하가 증가하기 때문이다. 이러한 현상은 시뮬레이션 모델의 결과만 가지고 네트워크 시스템을 설계하는 경우에 간과할 수 있는 현상이라고 할 수 있다.

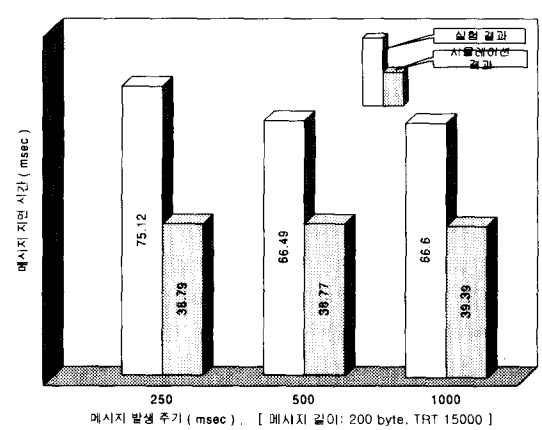


그림 12. 메시지 발생 주기 변화에 대한 메시지 지연시간.
Fig. 12. Message latency with respect to the change of message generation period.

메시지 길이의 변화

그림 13에는 각 노드에서 메시지 발생 주기가 500msec이고 TRT가 15000 bit time으로 설정된 경우에 메시지 길이의 변화가 메시지 지연 시간에 미치는 영향에 대한 결과가 주어져 있다. 그림에 나타난 바와 같이 메시지 길이가 증가할수록 시뮬레이션 결과와 실험 결과에서 모두 메시지 지연 시간이 증가하였다. 그러나 실험 결과에서 메시지 지연시간의 증가폭은 시뮬레이션 결과에 비하여 낮은 것으로 나타났다. 즉, 메시지 길이의 증가로 인한 데이터 링크 계층까지의 데이터 지연 시간의 증가가 응용 계층의 데이터 지연 시간에 미치는 영향이 크지 않음을 보여준다. 이는 응용 계층에서 메시지를 처리하는 시간이 균등하게 분포되지 않기 때문이며, 이러한 현상 역시 시뮬레이션 모델의 결과만 가지고 네트워크 시스템을 설계하는 경우에 간과할 수 있는 현상이라고 할 수 있다.

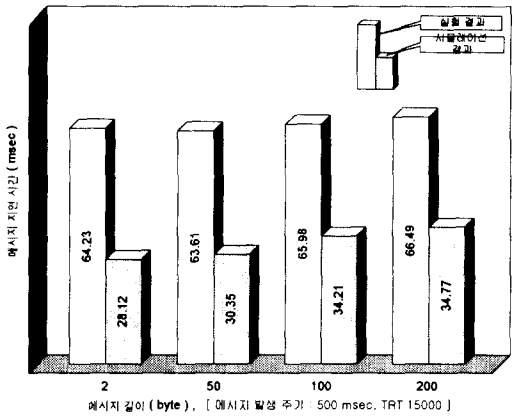


그림 13. 메시지 길이의 변화에 대한 메시지 지연시간.
Fig. 13. Message latency with respect to the change of message length.

TRT의 변화

그림 14에는 네트워크 내의 모든 노드에서 메시지 발생 주기가 500msec이고 생성되는 메시지의 길이가 2 바이트인 경우에 버스 파라미터인 TRT 값의 변화가 메시지 지연시간에 미치는 효과에 대한 결과가 나타나 있다. 여기서 발생되는 메시지의 우선 순위는 모두 낮은 우선 순위로 할당되었다. 그림에서와 같이 실험 결과와 시뮬레이션 결과에서 모두 TRT 값이 감소함에 따라 메시지 지연시간이 증가하였다. 이는 낮은 우선 순위의 메시지의 경우에 토큰이 네트워크 내의 모든 노드를 1회 방문하는데 소요되는 토큰 회전 시간이 TRT 보다 적은 경우에만 대기하는 데이터가 전송될 수 있기 때문이다. 그림에서 보는 바와 같이 TRT 값의 변화는 메시지 길이 또는 메시지 발생주기의 변화에 비하여 데이터 링크 계층과 응용 계층 모두의 메시지 지연시간에 미치는 영향이 매우 민감한 것을 알 수 있다.

실험 결과에 대한 고찰

이제까지 메시지 지연시간에 영향을 주는 네트워크 파라미터들에 대한 메시지 지연시간의 관계를 실험 모델과 시뮬레이션 모델을 통하여 살펴보았다. 이러한 실험 결과를 토대로 하여 Profibus를 이용한 자동화 시스템을 설계 및 구현하는 경우에 다음과 같은 사항을 고려하여야 할 것이다.

● 본 실험에서 사용된 응용 프로그램은 Write 서비스만을 수행하는 비교적 간단한 프로그램이었다. 그러나 이러한 간단한 작업의 경우에도 사용자 계층의 전체 메시지 지연시간이 데이터링크 계층까지의 지연시간에 비하여 상대적으로 크다는 것을 볼 수 있다. 특히 메시지가 자주 발생하는 경우에는 사용자 계층의 데이터 지연 시간이 크게 증가할 수 있다. 따라서 데이터 링크 계층까지만이 모델링 된 시뮬레이션 결과만을 가지고 네트워크 시스템을 설계하는 경우에는 응용 계층 및 사용자 계층에서 메시지 처리 시간에 대한 충분한 안전 계수(safety factor)를 고려하여야 할 필요가 있다.

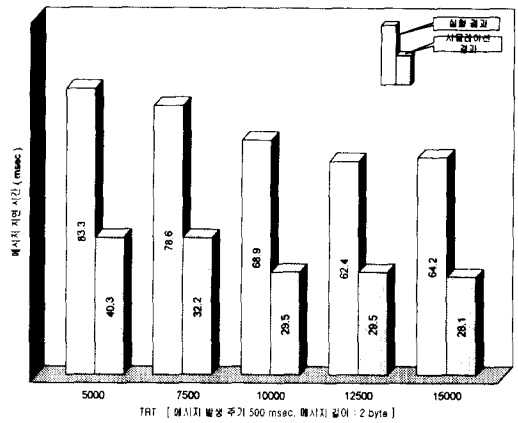


그림 14. TRT의 변화에 대한 메시지 지연 시간.
Fig. 14. Message latency with respect to the change of TRT.

● 메시지 발생 주기나 메시지 길이의 변화에 비하여 TRT 값의 변화는 데이터링크계층 뿐만이 아니라 사용자 계층까지를 고려한 메시지 지연시간에 대하여서도 매우 민감한 영향을 미친다. 따라서 실시간 데이터와 비실시간 데이터 트래픽이 공존하는 경우에 우선 순위가 높은 실시간 데이터가 우선적으로 전송되도록 하기 위하여 Profibus의 버스 파라미터인 TRT 값을 적절히 이용할 필요가 있다.

V. 결론 및 추후 연구 사항

본 연구에서는 현존하는 여러 종류의 필드버스들 가운데 이미 제품화가 완료되었고, 자동화 환경에서 요구하는 대부분의 통신 기능들을 제공하는 Profibus 프로토콜을 자동화 장비들 간에 실시간 통신을 지원하기 위한 네트워크 시스템으로 도입하는 경우에 필드버스 접속 소프트웨어의 구현 방법을 제시하였다. 본 연구를 통하여 구현된 Profibus 접속 소프트웨어는 산업용 컴퓨터로 널리 사용되고 있는 PC와 전용제어기인 IUC에서 동작될 수 있으며, 이러한 자동화 장비에서 여러 작업을 실시간으로 동시에 수행할 수 있도록 하기 위하여 CTask와 OS-9 등의 실시간 다중 작업 프로그래밍 환경 하에서 구현되었다. 본 논문에서 제시한 실시간 다중 작업 프로그래밍 환경 하에서 Profibus 접속 소프트웨어 구현 방법은 실제 산업 현장에서 Profibus를 이용한 자동화 시스템을 구축하는데 바로 사용될 수 있을 것이다.

본 논문에서는 구현된 Profibus 접속 소프트웨어를 이용하여 자동화 시스템에 적용될 수 있는 실험 모델을 구축하고, 이러한 실험 모델을 통하여 메시지 지연시간에 대한 네트워크 시스템의 성능 실험을 수행하였다. 성능 실험에서는 메시지 발생 주기, 메시지 길이 및 TRT(Target Rotation Timer) 등의 네트워크 파라미터 값의 변화에 대하여 사용자 계층에서의 메시지 지연시간을 측정하였으며 이러한 실험 결과를 Profibus의 데이터 링크 계층까지만이 모델링된 시뮬레이션 결과를 비교하여 데이터링크계층까지의 데이터 지연시간과 실제로 네트워크의 사용자가 경험하는 전체 메시지 지연시간과의 관계를 비교하였다. 또한, 데이터 링크

계층까지 만을 구현하는 시뮬레이션 모델을 이용한 네트워크 시스템 성능 해석에서 간과하기 쉬운 몇 가지 현상들을 실험 결과를 통하여 제시하였다. 본 연구의 결과에 의하면 사용자 및 응용 계층에서 발생하는 지연시간이 전체 메시지 지연시간에 상당한 부분을 차지하며, 따라서 Profibus를 이용한 자동화 시스템의 설계 및 구축시 이를 고려하여야 한다.

본 연구의 후속 연구로 본 연구를 통하여 개발된 Profibus 접속 소프트웨어를 이용하여 실제의 로봇, 컨베이어 벨트, NC선반과 PLC 및 각종 센서, 오퍼레이터 스테이션 등으로 구성된 첨단 생산 자동화 시스템의 실험 모델을 구축하는 연구가 진행되고 있다. 이러한 실험 모델은 생산 자동화 시스템에 필드버스를 도입하는데 필요한 기반 기술을 확보하고, 이를 실제 산업 현장의 생산 자동화 시스템에 적용하기 위한 방안을 제시하는데 활용될 것이다.

참고문헌

[1] R. Piementel, *Communication Networks for Manufacturing*, Prentice Hall, 1990.

[2] J. R. Jordan, *Serial Networked Field Instrumentation*, John Willey & Sons, 1995.

[3] *MAP 3.0 Specification 1993 Release*, World Federation of MAP/TOP Users Groups, 1993.

[4] A. H. McMillan and C. J. Gardner, *Mini-MAP '93*, Open I.T. Corp. 1994.

[5] *Basic Reference Model for Open System Interconnection*, ISO 7498, 1984.

[6] 홍승호, 김기암, 김지용, 고성준, "분산제어 및 자동화 시스템과 필드버스," 제어·자동화·시스템공학회지, 제2권 제4호, pp. 19-29, July, 1996.

[7] *InTech: Field Buses Special Issue*, ISA Publication, Nov., 1996.

[8] 홍승호, "필드버스 전개과정 및 발전전망," 월간 CONTROL, pp. 48-56, Sept., 1996.

[9] Seung Ho Hong, "Approximate analysis of timer-controlled priority scheme in the single- service token-passing systems," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 206-215, April 1994.

[10] K. Bender, *Profibus-The Fieldbus for Industrial Automation*, Carl Hanser Verlag & Prentice Hall, 1993.

[11] DIN 19 245 *Profibus Standard Part 1* : 1991.

[12] DIN 19 245 *Profibus Standard Part 2* : 1991.

[13] C. I'Anson and A. Pell, *Understanding OSI Applications*, Prentice Hall, 1993.

[14] G. Dickson and A. Lloyd, *Open Systems Interconnection*, Prentice Hall, 1992.

[15] M. Babb, "PCs: The foundation of open architecture control systems," *Control Engineering*, Jan., 1996.

[16] M. Babb, "PC-Based control : will the alternative become the standard?," *Control Engineering*, Nov., 1996.

[17] T. Wagner, *CTask : A Multitasking Kernel for C*, Ferrari Electronic GmbH, 1990.

[18] *Profibus Communication Interface Layer7 for CP5412-A1 Controller*, SOFTING GmbH, 1994.

[19] *IUC User's Manual*, PEP Modular Computers, 1993.

[20] *MC68302 User Manual*, Motorola.

[21] P. Dibble, *An Advanced Programmers Guide to OS-9*, Microware Systems, 1994.

[22] M. Heilpern, *The OS-9 Primer*, Microware Systems, 1994.

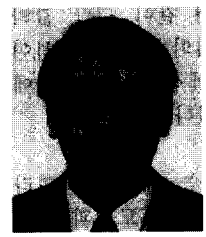
[23] A. Ray, S. H. Hong, S. Lee and P. J. Egbelu, "Discrete-event/continuous-time simulation of distributed data communication and control systems," *Trans. of the Society for Computer Simulation*, vol. 5, pp. 71-86, Jan., 1988.

[24] Seung Ho Hong, "Scheduling algorithm of data sampling times in the integrated communication and control systems," *IEEE Trans. on Control Systems Technology*, vol. 3, pp. 225-230, June, 1995.

[25] C. D. Pegden, R. E. Shannon and R. P. Sadowski, *Introduction to Simulation Using SIMAN*, McGraw-Hill, 1995.

김 기 암

1972년 11월 26일생. 1995년 한양대 제어계측공학과 졸업. 동대학원 석사 (1997). 1997년~현재 현대정보기술(주) 산전정보기술연구소 연구원. 주관심분야는 분산제어 및 필드버스임.



홍 승 호

1956년 5월 31일생. 1982년 연세대 기계공학과 졸업, 텍사스공대 공학석사 (1985), 펜실베이니아주립대 공학박사(1989), 1989년~1992년 한국전자통신 연구소 선임연구원, 1992년~현재 한양대 제어계측공학과 부교수. 주관심분야는 분산제어/자동화 통신망임.