1999년도 대한전기학회 하계학술대회 논문집 1999. 7.19-21

지능형 센서용 필드버스 데이터링크계층 프로토콜 설계 및 구현

김유철, 홍승호
한양대학교 제어계측공학과

Design and Implementation of Fieldbus Data Link Layer Protocol for Intelligent Sensor

Yu Chul Kim, Seung Ho Hong
Control & Instrument Engineering in Hanyang University

Abstract - 첨단의 자동화 시스템을 구축하기 위해서는 각 필드기기에서 생산되는 정보들을 적절한 형태로 가공하여 적시에 필요한 공정으로 제공하여 줄 수 있는 지능형 센서 및 필드기기의 도입이 필요하다. 이러한 필드기기들이 유기적으로 정보를 교환하고 공유하기 위해서는 통신망 시스템을 구축할 필요가 있다. 필드버스는 자동화 및 분산 제어 시스템의 컴퓨터 통신망 계층구조에서 최하위 계층 기기들 간에 실시간 통신을 제공하는 산업용 통신망이다. 본 연구에서는 통신용 프로세서인 MC68360을 기반으로 하여 필드버스의 일종인 Profibus의 물리계층과 데이터링크계층 프로토콜을 구현하였다. 물리계층은 프로세서의 UART 통신 기능과 RS-485칩을 사용하여 구현하고, 데이터링크계층 프로토콜은 프레임 분석과 송수신, 에러처리, 흐름제어, 매체접속권한 관리 등의 기능을 소프트웨어로 구현하였다. 또한 지능형 센서 본래의 목적중의 하나인 원격관리 기능을 위하여 각 필드기기의 노드 주소, 타이머 값 등의 통신 파라미터를 원격 마스터에서 설정할 수 있도록 관리계층의 기능을 추가하였다. 본 연구에서는 각각 하나의 노드기능을 담당하는 여러 개의 보드들로 구성된 testbed를 구축하고, 다양한 통신환경에서 초기화, 정상, 비정상 상태 등의 동작을 실험하였으며, 이를 통하여 지능형 센서용 필드버스의 데이터링크계층 프로토콜이 정상적으로 동작됨을 확인하였다.

1. 서 론

본 연구에서는 Profibus의 물리계층과 데이터링크계층 프로토콜을 구현하였다. Profibus의 물리계층은 NRZ(None Return to Zero) 인코딩, BUS topology의 특성을 가지며 MC68360 프로세서의 UART 통신 기능과 RS-485칩을 사용하여 구현하였다. 그리고 데이터링크계층 프로토콜은 프레임 분석과 송수신, 에러처리, 흐름제어, 매체접속권한 관리 기능을 소프트웨어로 구현하였다. 구현된 하드웨어 및 소프트웨어의 동작을 검증하기 위하여 여러 개의 노드를 제작하여 네트워크를 구성하고 다양한 통신환경에서 초기화, 정상, 비정상 상태의 동작을 실험하였다.

2. 본 론

2.1 데이터 연결계층 및 물리계층의 하드웨어구조

본 연구에서 PROFIBUS의 데이터 연결 계층을 구현하기 위해서 MOTOROLA사의 MC68360 CPU를 장착한 보드를 제작하였다. MC68360은 내부에 CPU32+ 코어를 가지고 있으며 별도로 통신을 위한 COMMUNICATION PROCESSOR를 가지고 있다. 제작한 보드에는 처음 시동프로그램을 저장하기 위한 1M비트 플래시 롬과 프로그램 실행 시 필요한 1M비트 S램을 4개 사용하고, PC와의 인터페이스를 위한 듀얼포트램을 장착하였다. 그리고 RS-485칩(MAX1487)을 사용하여 네트워크에 접속하며 모니터링을 위해서 RS-232칩(MAX233CPP)을 추가로 장착하였다.

2.1.1 인터럽트 제어

MC68360의 인터럽트 제어는 CPM의 인터럽트 소스를 관리하는 CPIC (CPM INTERRUPT CONTROLLER)와 SIM60 인터럽트 제어기, 그리고 전체적으로 인터럽트를 관리하는 로직이 서로 유기적으로 연결되어 관리하게 된다. MC68360에서 최대로 정의 할 수 있는 인터럽트의 개수는 256개이며 이중 64개는 프로세서에 의해 정의되어지는 부분이다. 인터럽트의 레벨은 Level 1에서 Level 7까지로 나뉘며 Level 7은 Non Maskable이고 우선 순위가 가장 높다. 이중에서 CPM은 Level 4로 설정하였다. 본 연구에서 사용자가 정의하는 인터럽트 소스는 다음과 같다.

표 1. 사용자 정의 인터럽트 소스

인터럽트 소스	용 도
SCC 4	RS-485 수신 인터럽트
SCC 2	RS-232 수신 인터럽트
Timer 1	TRT(Token Rotation Timer)
Timer 2	Time-Out-Timer
Timer 3	Slot Timer
Timer 4	Sync Interval Timer
PortC 11	DP-RAM INTERRUPT (PC INTERFACE)

위의 인터럽트 소스들은 CPM 인터럽트 제어기(CPIC)에서 관리하며 우선순위에 따라 CPU에 인터럽트를 요청하게 된다. CPIC의 설정은 CICR(CPM Interrupt Configuration Register)에서 하게 된다. CICR은 SCC간의 우선순위를 결정하고 어떤 Level로 CPU에 인터럽트를 요청할 것인지를 결정한다.

2.1.2 직렬 통신 제어기

MC68360에서는 4개의 SCC를 통해 직렬 통신을 제공한다. 각각의 SCC는 독립적으로 동작하며 별도의 보드레이트 생성기로부터 통신 속도를 제공받는다. SCC의 설정은 32비트 길이의 레지스터(General SCC Mode Register : GSMR)를 통해 하게 되며 여기에서 설정하게 되는 값은 CRC 종류, sync 비트 길이, 인코딩 종류, 프리앰블 종류 등의 물리계층에 필요한 파라미터들이다. 그리고 범용으로 사용되는 프로토콜의 경우 프로토콜 타입을 설정해주도록 되어 있다. RS-485와 RS-232의 경우에는 UART를 지정해주면 된다.

PROFIBUS 표준안 Part 1,2[2]에서는 9600, 19200, 93750, 187500, 500000 bps를 지원하도록 하고 있다. MC68360에서는 BRG Configuration

* 본 연구는 과학재단 특정 연구과제 연구비 지원에 의한 결과임 (과제번호 : 97-01-00-11-01-3)

Register에서 보드레이트를 설정해주며 내부 클럭이나 외부클럭 모두 사용할 수 있다. 보드레이트를 구하는 식은 다음과 같다.

Baudrate=system clock(33MHz)÷DIV16÷DIV

DIV16은 시스템 클럭을 16분주하여 사용할 것인지를 지정하고 DIV값에 따라 통신 속도가 결정된다.

2.1.3 타이머

MC68360은 4개의 16비트 General Purpose Timer와 16개의 RISC Timer를 제공한다. General Purpose Timer는 2개를 연결하여 32비트 타이머로 사용할 수 있으며 업 카운터를 계속 증가시키다가 설정값이 되면 인터럽트를 발생시키도록 되어 있다.

타이머를 사용하기 위해서는 TGCR(Timer Global Configuration Register), TMR(Timer Mode Register), TRR(Timer Reference Register)을 설정해주어야 한다. TGCR은 타이머를 초기화하는데 사용되며 cascade mode를 사용할 것인지도 설정해주어야 한다. TMR은 타이머의 정밀도와 클럭의 종류를 지정하는데 사용되고 TRR은 인터럽트가 발생되는 설정 값을 지정하도록 되어 있다.

PROFIBUS FDL에서 필요로 하는 타이머는 Slot Timer, TTO(Time-out Timer), Sync Interval Timer, TRT(Token Rotation Timer)이며 각 타이머마다 하나씩을 할당하였다.

Slot Time은 요구 프레임을 전송한 후에 응답 프레임이 오기까지의 시간의 최대값을 지정하는 시간이고, Time-out Timer는 초기화시 또는 토큰 상실시에 낮은 주소로부터 토큰을 발생시킬 수 있게 한다. Sync Interval Timer는 전송기 불량으로 인한 연속적인 신호 발생을 금지시키는 역할을 수행하며 TRT는 매체의 트래픽에 따라 전송권한을 갖는 시간을 조절할 수 있게 해준다. 다음 표는 실제로 계산된 타이머들의 값이다.

표 2. 보드레이트에 따른 각 타이머의 설정 값
(단위 : bit times)

Time Parameter	9600 bps	19200 bps	93750 bps	187500 bps	500000 bps
T_SYN	33	33	33	33	33
T_SYNI	11385	11385	11385	11385	11385
T_TD	0.05	0.09	0.487	0.9375	2.5
T_SET	0.29	0.58	2.95	5.681	15.151
max_T_SDR	2.32	4.65	23.63	45.454	121.21
min_T_SDR	1.16	2.33	11.81	22.727	60.606
T_ID1	35.58	36.16	40.90	46.36	80.814
T_ID2	35.58	36.16	40.90	46.36	121.21
T_SL1	16.01	19.01	43.52	71.69	169.51
T_SL2	49.26	50.52	60.79	72.60	129.11
T_SL	49.26	50.52	60.79	72.60	169.51
T_TO	394.077	404.154	486.34	580.81	1356.12
TRT	28542	28555	28657	28776	29745

2.2 데이터연결계층의 소프트웨어

데이터연결계층에는 매체접근제어기(MAC: Medium Access Control)와 논리적연결제어기(LLC: Logical Link Control)가 있다. 매체접근제어기는 매체를 이용하고 있는 노드 중에서 오직 하나의 노드만이 데이터를 전송할 권한을 가지도록 제어하는 역할을 하며, 논리적연결제어기는 통신을 하려는 노드들간에 논리적 연결을 설정하고 해제하는 기능을 비롯하여 노드들간의 통신과정에서 발생할 수 있는 각종 오류를 처리함으로써 노드간에 신뢰할만한 데이터 전송이 이루어지도록 하는 역할을 수행한다. FDL(Fieldbus Data Link Layer)은 FDL User에게 SRD(Send Data with Request Data), SDA(Send Data with Ack), SDN(Send Data with No Ack), CSRD(Cyclic Send Data with Request Data)의 서비스를 제공한다. 각각의 서비스에 맞는 알고리듬에 따라 프레임을 생성, 전송하고 매체로부터 수신되는 프레임을 분석하여 자신이 목적지인 프레임만을 선별 및 처리하여 LLI(Lower Layer Interface)로 올려 보낸다. 그리고 노드들의 주소록(LAS) 작성 관리, 노드 추가를 위한 GAP Update, 전송권한(Token) 관리 등의 기능도 수행한다.

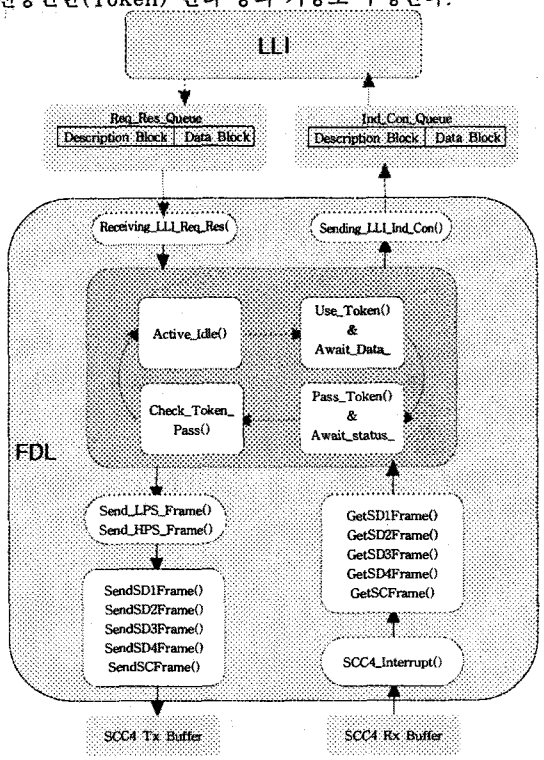

그림 1. FDL 소프트웨어 구조 및 인터페이스

프로그램의 동작은 전송 및 수신으로 나눌 수 있다. 전송은 LLI로부터의 큐 입력, 조건에 알맞은 전송프레임 생성, SCC의 UART 기능을 이용한 프레임 전송의 순으로 처리되며, 수신은 SCC 인터럽트 발생시 UART 수신 버퍼로부터 수신 큐에 복사, 큐로부터 프레임 수신, 프레임 에러처리, 프레임 분석, 주소 및 프레임 종류에 따른 처리 및 응답 프레임 전송, 상위계층으로 보고해야할 경우 LLI로의 큐 입력 순으로 처리된다. FDL 내부에서는 11가지의 상태의 천이를 통해 각각의 상황에 맞는 처리를 하게 된다.

FDL에는 High와 Low 두가지의 메시지 우선순위가 있으며 알람등의 이벤트는 High 우선순위, 일반 데이터는 Low 우선순위로 분류된다. 이들 메시지는 서로 다른 큐를 사용하며 우선 High 우선순위 메시지를 처리하고 난 후에 THT(Token Holding Time)을 검사하여 남은 시간동안 Low 우선순위 메시지를 처리하게 된다.

표 3. LLI 인터페이스 함수

함수명	설 명
Receiving_LLI_Req_Res()	LLI Interface queue에서 request 또는 response data를 읽어오는 함수
Sending_LLI_Ind_Con()	LLI Interface queue로 indication 또는 confirm을 보내주는 함수

표 4. 초기화 및 상태함수

함수명	설명
main()	전체적인 태스크와 상태함수들을 호출한다.
App()	FDL User Program
Register LAS()	LAS를 등록한다.
FindNS()	Next Station의 Address를 찾는다.
get_trt()	TRT Timer 값을 읽어와서 hold time을 계산한다.
Offline()	FDL parameter 및 시스템 파라미터를 초기화한다.
Listen_ Token()	주소록(LAS table)을 작성하고 논리적인 토큰링에 참여하기 위한 준비 상태함수
Active_Idle()	토큰이나 요구 프레임을 기다리다가 그에 알맞는 응답을 해주는 상태함수
Claim_ Token()	타이머가 다 소모되도록 프레임이 오가지 않는 경우 다시 논리적인 토큰링을 초기화하는 작업을 수행하는 상태함수
Use_Token()	자신이 전송권한을 가지고 요구프레임을 전송하는 상태함수
Await_Data_ Response()	전송한 요구프레임에 대한 올바른 응답 프레임이 올 때까지 기다리는 상태함수
Check_Access Time()	자신이 사용할 수 있는 시간을 계산하는 상태함수
Pass_Token()	다음 스테이션으로 토큰을 넘기는 역할을 수행하는 상태함수
Gap_Update()	GAP Update를 수행하는 상태함수
Check_Token_ Pass()	토큰이 올바르게 전송되었는지 모니터링을 하는 상태함수
Await_Status Response()	GAP_Update 작업을 수행할 때 요구 프레임을 전송하고 나서 기다리는 상태함수
Passive_Idle()	슬레이브 노드로 지정되어 있을 때 자신에게 수신된 요구 프레임에 대해 알맞은 응답 프레임을 전송하는 상태함수

표 5. 프레임 처리 함수

함수명	설명
GetSD1 Frame()	SD1 프레임을 수신하여 분석하고 구조체에 복사하는 함수
GetSD2 Frame()	SD2 프레임을 수신하여 분석하고 구조체에 복사하는 함수
GetSD3 Frame()	SD3 프레임을 수신하여 분석하고 구조체에 복사하는 함수
GetSD4 Frame()	SD4 프레임을 수신하여 분석하고 구조체에 복사하는 함수
GetSC Frame()	SC 프레임을 수신하여 분석하고 구조체에 복사하는 함수
SendSD1 Frame()	SD1 프레임을 구성하고 전송하는 함수
SendSD2 Frame()	SD2 프레임을 구성하고 전송하는 함수
SendSD3 Frame()	SD3 프레임을 구성하고 전송하는 함수
SendSD4 Frame()	SD4 프레임을 구성하고 전송하는 함수
SendSC Frame()	SC 프레임을 구성하고 전송하는 함수
Send_HPS_ Frame()	High Priority의 프레임을 전송하는 함수
Send_LPS_ Frame()	Low Priority의 프레임을 전송하는 함수
CheckSD()	프레임의 SD(Start Delimiter)를 체크하여 프레임의 종류를 판별하는 함수
getch()	RS-232 버퍼 값을 읽어들이는 함수
putch()	RS-232 버퍼에 값을 넣어 전송을 요구하는 함수

2.3 동작 상태 검증

PROFIBUS 프로토콜이 정상적으로 동작하기 위해서는 두 개 이상의 노드와 프로토콜 계층 전부가 구현되어야 한다. 하지만 본 연구에서는 하위계층인 데이터연결계층과 물리계층만을 구현하였으므로 타이머 인터럽트를 이용하여 가상의 메시지를 생성하고 이를 FDL 큐에 입력되도록 하였다. 실험은 PROFIBUS Standard에 지정되어있는 9600, 19200, 93750, 187500, 500000 (bps)의 전송속도에서 진행되었다. 데이터연결계층의 테스트는 토큰의 전달 및 데이터의 송수신 등 정상상태의 동작뿐만 아니라 여러 가지 예외상황에서도 제대로 동작하는지 검증하였다. PROFIBUS에서 생성되는 에러와 예외 상황의 종류는 아래와 같다

① 여러 개의 토큰이 생성된 경우
② 토큰을 잃어버린 경우
③ 토큰이 전달 도중에 사라진 경우
④ 여러 스테이션이 같은 주소를 가지고 있는 경우
⑤ 스테이션에 잘못된 전송기와 수신기를 사용한 경우
⑥ 다른 스테이션이 논리적인 토큰링에 삽입되거나 삭제되는 경우

각 상황마다의 에러처리는 FDL State Machine에서 수행하게 되며 본 연구를 통하여 개발된 프로그램에서는 FDL_state.error 변수에 그 상태에서 발생한 에러의 종류를 기입하여 다음 상태에서 그 에러에 알맞은 처리를 하도록 구현하였다.

검증방법으로는 PC 인터페이스 기능을 통한 데이터 확인 방법과 Protocol Analyzer를 이용한 프레임 확인 방법, 그리고 CPU 디버그 장비를 이용한 프로그램 내부 동작상태 확인 방법을 사용하였다.

3. 결 론

본 연구에서는 Profibus의 물리계층과 데이터링크계층 프로토콜을 하드웨어 및 소프트웨어를 이용하여 Embeded System을 구현하였다. 여러 개의 Master와 Slave 노드를 만들어서 네트워크를 구축하고 각 전송속도에서 보드의 동작상태를 확인하였다. 그리고 노드의 추가 및 제거의 경우에도 프로토콜의 예외처리가 이상 없이 동작됨을 확인하였다. 본 연구를 통하여 구현된 Profibus 보드는 Dual Ported RAM을 가지고 있으며 PC 슬롯에 장착할 수 있게 설계되었다. 이로 인해 PC에서 네트워크 모니터링이 가능하고 PC가 Master 노드로서 Supervisor 역할을 수행할 수도 있다. 또한 CPU의 입출력 포트를 외부로 연결하여 센서나 액츄에이터를 부착할 수 있도록 하였다.

추후에 LLI, FMS, FB(Function Block)등의 상위계층을 추가로 구현하고 센서, 제어기, 액츄에이터를 포함한 실제 공정을 구축하여 제어루프의 동작상태 및 메시지 지연시간을 테스트할 예정이며 네트워크 자원의 효율적인 이용을 위하여 대역폭 할당기법을 FDL 내에 추가로 구현할 예정이다.

[참 고 문 헌]

[1] DIN 19 245 Profibus Standard Part 1, 1991
[2] DIN 19 245 Profibus Standard Part 2, 1991
[3] 김지용,홍승호, "PROFIBUS에서 대역폭 할당기법 구현", 한국 자동제어 학술회의 논문집, 1권, pp.97-100, 1997
[4] MOTOROLA, MC68360 - Quad Integrated Communications Controller User's Manual, Motorola
[5] K.Bender, Profibus - The Fieldbus for Industrial Automation, Carl Hanser Verlag & Prentice Hall, 1993