제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호 (2010. 4)

데이터그램과 큐를 이용한 EIA-485 통신

김두호*, 이진*, 이혁준*, 신윤호*, 김정선*
*한양대학교 컴퓨터공학과
e-mail:fr2air@hanyang.ac.kr, babytaiji@hanyang.ac.kr, breeze-45@hanmail.net,
iashinyh@hanyang.ac.kr, kimjs@hanyang.ac.kr

EIA-485 Communication using Datagram & Queue

Dooho Kim*, Jin Lee*, Hyuk-Joon Lee*, Yunho Shin*, Jung Sun Kim*
*Dept of Computer Science & Engineering, Han-yang University

요 약

임베디드 시스템의 유선 네트워크 중 저비용의 EIA-485 네트워크에서 큐(Queue)와 데이터그램(Datagram)을 이용하여 Ethernet의 장점인 흐름제어를 소프트웨어적으로 구현 가능한 방법을 제시하고, Ethernet의 또 다른 장점인 에러정책에 있어서 EIA-485 통신 방식에서 부족한 에러정책을 에러 예방의 차원에서 소프트웨어적인 구현 방법을 제안해 본다.

1. 서론

최근 임베디드 시스템의 네트워크 구성을 위해서 여러 데이터 통신 방식들이 사용되어지고 있다. 이런 데이터 통식 방식은 LAN, 시리얼 통신과 같은 물리적인 케이블을 필요로 하는 유선 통신과 케이블을 필요치 않는 RFID, 블루투스, WLAN 등의 무선 통신으로 구분 할 수 있다. 근래에는 무선 통신 방식이 많이 각광을 받고 있지만, 보안상의 문제점이나 통신 거리상의 문제로 아직은 유선 통신 방식이 많이 사용되어 지고 있다.

임베디드 시스템을 구축에 있어서 클라이언트의 요구사항 및 환경적인 제약 사항 등이 많은 부분을 차지하겠지만 비용적인 면이 가장 큰 관심 사항일 것이다. 이런 비용적인 면을 고려한다면 LAN보다는 시리얼 통신을 채택하는 것이 올바른 선택일 것이다.

하지만 LAN은 흐름제어, 에러검출, 광범위한 통신거리, 속도 등과 같은 많은 장점을 가지고 있다. 이러한 면모에서는 시리얼 통신이 비약한 면을 보이지만 시러얼 통신 규격 중에서 EIA-485 통신을 이용하면 통신 거리, 속도등의 문제를 해결할 수 있다. 그러나 에러검출 및 흐름제어에 문제점을 아직도 내포하고 있다.

본 논문에서는 이러한 문제점을 다소 해결할 수 있는 데이터그램(Datagram)과 큐(Queue)를 이용한 EIA-485 통신 방식을 제안한다.

2절에서는 EIA-485 통신 프로토콜에 대해서 논하고, 3절에서는 데이터그램을 이용한 데이터 식별에 대해서 설명한다. 그리고 4절에서는 수신 Interrupt 방식을 이용하면 발생하는 문제점에 대해서 알아보고, 5절에서는 이 문제점을 해결하기 위한 방법을 제시해 본다. 마지막으로 6절에서는 결론 및 향후 과제에 대해서 논해본다.

* 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. 20100000215)

2. EIA-485 통신 프로토콜

본 논문에서 다루고자 하는 EIA-485 통신 프로토콜 규격[1]의 하드웨어적인 특징과 LAN의 특징을 아래의 <표 1>에서 나타내고 있다. 여기서 LAN은 Ethernet인 TCP/IP로 한정하고, 여러 Ethernet 중에서 EIA-485와 가장 비슷한 10BASE-T[2]와 비교한다.

<표 1> RS-485와 LAN 통신 규격 비교

Specification	EIA-485	10BASE-T
cable length, max	1.2km	1km
Data rate, max(bps)	10M	10M
Transmission system	Half Duplex	Full Duplex

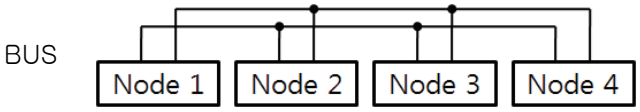
10BASE-T를 이용한 LAN의 경우 Full Duplex를 위해서 1:1의 통신을 하게 되지만 중간에 허브를 이용하여 네트워크의 구성이 가능하다. 그러나 EIA-485 통신 방식은 최대 Driver와 Receiver 수가 각각 32개나 가능하기 때문에 모든 장치들을 동일 라인에서 데이터 전송 및 수신을 할 수 있는 BUS형태의 네트워크를 구성할 수 있다.

EIA-485 통신 방식은 BUS 형태의 네트워크를 구성하므로 동시에 여러 개의 마스터가 출력을 하여 데이터 충돌현상을 야기 할 수 있는 상황을 Non-Echo Mode와 Echo-Mode 두 가지의 Mode를 제공하여 데이터 흐름 제어를 하여 해소하고 있다.

Echo Mode 통신방식에서 RxD 라인은 항상 Multi-Point BUS에 접속되어있어야 하며, TxD 라인은 데이터를 출력할 때만 Multi-Point BUS에 접속하여야 한다.

Non-Echo Mode 통신 방식은 TxD 라인을 Multi-Point BUS에 접속시키면 RxD 라인은 버스에서 단락시키고, TxD 라인이 Multi-Point BUS에서 단락되면 RxD 라인은 버스에 접속하는 방식이다.[3]

EIA-485 통신 방식에서 BUS 형태의 네트워크를 구성하여 데이터의 송수신시에 데이터의 식별은 소프트웨어적인 방법으로 해결을 하여야 하며, 이때의 데이터 식별을 데이터그램[4]을 이용하였다.



(그림 1) EIA-485구성하는 BUS형태의 네트워크

3. 데이터그램을 이용한 데이터 식별

[4]에서는 데이터 식별을 위해서 Ethernet Bus 통식 방식을 응용한 소프트웨어적인 데이터그램을 제시하고 있다. 데이터그램의 형태는 아래 (그림 2)와 같다.

Header[5]				Data[N]
DE[1]	SR[1]	OpId[1]	Length[2]	Data[N]

(그림 2) 데이터 패킷화를 위한 데이터그램

본 논문에서는 1byte의 데이터 통신을 하는 시리얼 통신에서의 원활하게 데이터그램의 시작점을 식별할 수 있게 (그림 2)의 구조를 아래 (그림 3)과 같이 변경하였다.

Header[6]					Data[N]
St[1]	DE[1]	SR[1]	OpId[1]	Length[2]	Data[N]

(그림 3) 변경된 데이터그램

한번에 1byte씩 데이터 통신을 하는 시리얼 통신에서 N byte로 이루어진 데이터 그램을 수신할 때 1byte의 St (Start byte)를 이용한다면, St를 수신한 이후에 자신에게 오는 데이터인가 아닌가를 식별하는 작업이 훨씬 수월하게 이루어질 수 있다.

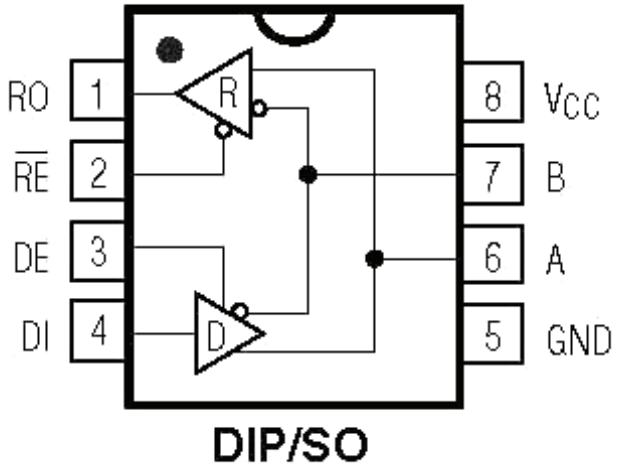
그리고 [4]에서 제시하는 DE (Destination Address)와 SR(Source Address)를 통해서 각 장치들 간의 데이터의 식별이 가능하다. <표 2>는 데이터그램을 구조체로 정의하고 있다.

<표 2> Structure of Datagram

Syntax	Bytes
struct SerialData {	
Byte StartByte;;	1
Byte DestinationAdress;	1
Byte SourceAddress;	1
Byte OperationId;	1
Word Length;	2
Var Data;	N
}	

4. 수신 Interrupt 방식의 문제점

EIA-485 통신 방식은 Transmission system이 Half Duplex로 이루어져 있는 관계로 한 번에 한 장치에서만 데이터 송신이 가능하다. 데이터를 송신하기 위해서는 아래의 (그림 4)[5]에서 보이는 $\overline{RE}$ Pin과 DE Pin을 Enable하면 송신을 할 수 있는 상태가 된다. 이러한 방식이 Non-Echo Mode 통신 방식이며, 데이터의 안전성을 위해서 Echo Mode 보다는 Non-Echo Mode 이용해야 한다.



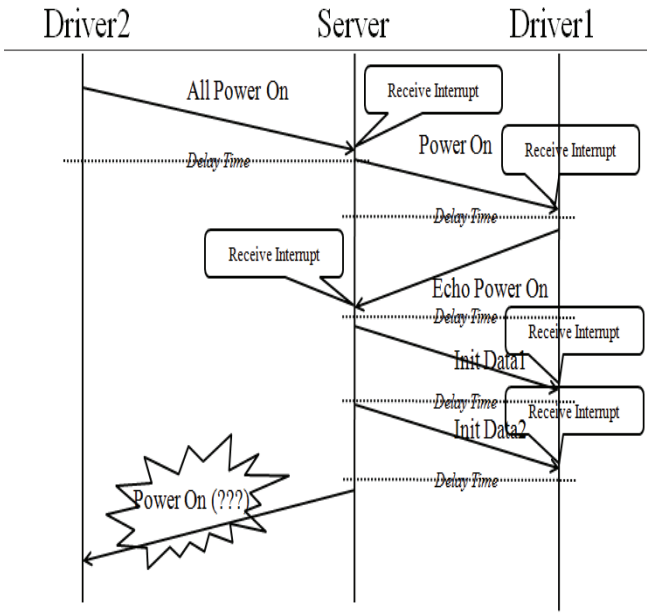
(그림 4) EIA-485 Driver

$\overline{RE}$ Pin과 DE Pin을 Enable시킨 후 Disable할 때 약간의 Delay Time을 필요로 하게 되는데, 만약 데이터를 송신한 후 Delay Time을 가지지 않고 Disable을 시키면 Receiver 측에서 데이터 수신을 완료하기 전에 데이터 송신이 Off 시키게 된다. 이러한 현상은 비정상적인 데이터가 수신을 발생시킨다. 그러므로 약간의 Delay Time이 주어져야 하는 것이다. 하지만 이 Delay Time 때문에 데이터 통신에 문제가 발생할 수 있다. 이 문제점에 대해서는 후에 기술하도록 한다.

항상 송신한 데이터가 안전하게 전송이 되었는지 확인하기 위해서는 소프트웨어적인 Echo Mode를 이용해야 한

다. 그러나 소프트웨어적인 Echo Mode를 이용하면 문제가 발생할 확률이 적어지지만 송신측에서 데이터를 보내고 Delay Time을 가지고 다시 수신측에서 Echo하기 위해서 데이터를 송신한다면 두 배의 Delay Time을 가져야 하기 때문에 성능적인 면에서 떨어질 수 밖에 없다. 그러므로 송신측에서는 수신한 데이터가 수신이 잘 되었는지 꼭 확인해야 되는 데이터만 소프트웨어적인 Echo Mode를 이용해야한다.

그리고 MCU에서 Multi Tasking과 같은 환경을 만들기 위해서 데이터 수신을 Interrupt 방식을 많이 이용하고 있다. 그러나 Interrupt 수신 방식과 Non-Echo Mode, 소프트웨어적인 Echo Mode를 동시에 사용하게 되면 문제가 발생한다.



(그림 5) 문제 상황

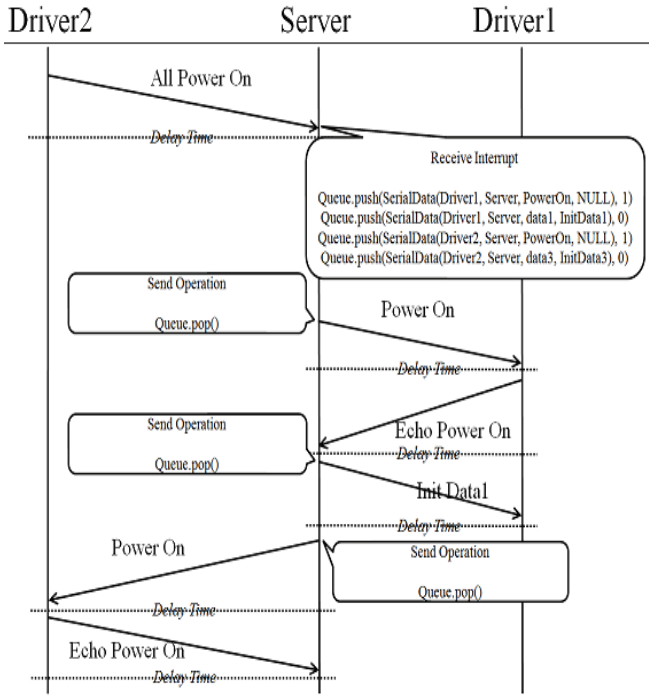
데이터를 저장하고 중앙에서 명령을 내리는 Server Driver가 있고, Server Driver에서 데이터를 가져오고, 명령을 받는 Driver1, 2가 있다. Driver2에서 Server Driver에게 전체 Driver들의 전원을 켜달라는 요청을 하면 Server Driver는 연결된 전체 Driver들에게 전원을 켜라는 명령을 내려야한다. 전원 On/Off와 같은 중요한 명령은 데이터의 누실이 없이 안전하게 송수신되어야 하므로 소프트웨어적인 Echo Mode를 이용하여 각 Driver들은 받은 Power On 메세지를 Server Driver에게 Echo한다. All Power On 요청을 받은 Server Driver에서는 오름차순으로 Power On명령을 Driver들에게 하달한다. 처음 Driver1에게 Power On명령을 내리게 되면 안전하게 명령이 잘 전될 되었는지 Echo를 기다린다. Power On명령을 받은 Driver1은 Power On 메시지를 Server Driver에게 Echo를 해주면 이를 받은 Server Driver은 전원이 켜졌을 때 필요한 초기 데이터들을 Driver1에게 보내준다. 이때의 데이터는 Non-Echo Mode로 송신을 하게 된다. 모든 데이터가 보내진 후 다음 차례인 Driver2에게 Power On명령을 내려야 한다.

그러나 Interrupt 수신 방식에서는 시리얼하게 전송되어지는 데이터가 1byte가 채워지면 수행 중이던 Operation은 Stack에 저장을 하고 Interrupt 영역으로 이동하게 된다. 그 후 Interrupt 영역내의 모든 역할을 마치게 되면 다시 수행 중이던 Operation으로 돌아오게 되는데 이점이 바로 문제가 된다. 최초의 Driver2에서 요청한 메시지를 어딘가에 저장하고 있지 않다면 Driver1의 Echo 메시지 때문에 다시 한번 수신 Interrupt가 발생을 하게 되고, 이전의 All Power On 요청은 사라지게 되는 것이다. 막상 Driver1에게 처리해줘야 할 모든 작업들을 마치고 Driver2에게도 Power On 명령을 내리려고 해도 (그림 5)에서 보이는 것처럼 어떤 명령을 내려야 하는지 알 수가 없게 되는 것이다.

5. 큐(Queue)를 이용한 흐름 제어

4절에서 다루었던 문제점을 큐(Queue)를 이용한 방법으로 해결해 보려 한다.

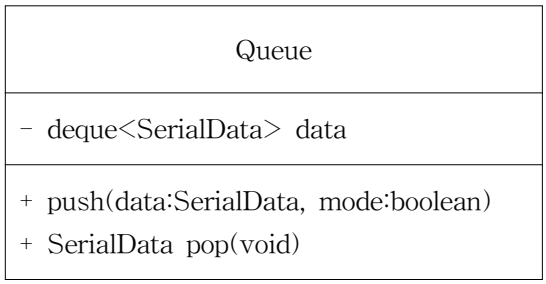
데이터를 수신하게 되면 수신 받은 메시지를 통해서 다시 다른 Driver에게 메시지를 전달해야 한다면 그 전달되야 하는 메시지를 큐(Queue)에 임시적으로 저장을 해두는 것이다. 또한 메시지들에게 중요도를 부여하여 우선순위에 맞게 메시지를 전달하여 시리얼 통신의 흐름을 제어하는 것이다.



(그림 6) Queue를 이용한 흐름제어

(그림 5)의 문제 상황을 Queue를 이용하여 (그림 6)과

같이 해결하였다. 처음 All Power On 요청 메시지를 Driver2에게 받게 되면 오름차순으로 Driver1부터 Power On 명령을 큐에 저장을 하며, 이 때 데이터그램 단위로 저장을 하게 된다. 또한 마지막 파라미터 값으로 1, 0으로 식별하여 소프트웨어적인 Echo Mode가 필요로 하는지 아닌지를 판별한다. 모든 메시지를 큐에 저장해둔 다음 나머지 작업을 마치고 Interrupt 상황을 탈출한 다음 FIFO의 개념으로 Queue의 데이터그램을 각 Driver에게 송신한다. 이렇게 하면 어떠한 메시지를 어느 곳으로 보내야 할지가 명확해진다. 흐름제어가 명확해 지기 때문에 하드웨어적으로 에러검출을 못해주는 부분에 있어서 예방을 할 수 있게 되는 것이다. 아래는 큐(Queue)를 UML의 형태로 표기한 그림이다.



(그림 6) C++ Template 이용한 Queue

C++ Template의 deque을 이용하여 Queue를 구성하였으며, push() 메소드에서는 데이터 저장을 Vector의 push_back() 메소드에게 위임하고, pop() 메소드에서는 데이터의 꺼냄을 pop_front() 메소드에게 위임하여, 처리하고 있다.

6. 결론 및 향후 과제

본 논문에서 제시한 EIA-485 통신 방식을 데이터그램과 큐를 이용하여 데이터 흐름을 제어하여 에러발생에 대한 미연의 방지를 이룰 수 있는 장점을 볼 수 있다. 그리고 데이터의 흐름을 제어한다는 것은 이전 3절에서 거론하였던 Delay Time에 의한 불안전한 데이터 전송 또한 방지할 수 있다는 것 또한 장점으로 들 수 있다.

앞으로 EIA-485 통신을 임베디드 시스템에서 부담없이 사용할 수 있는 1세대 Ethernet 통신과 비슷한 성능을 내기 위해서는 에러를 예방하는 차원이 아닌 검출도 해낼 수 있는 방법 구상이 향후의 과제이다.

참고문헌

[1] Comparing EIA-485 and EIA-422-A Line Drivers and Receivers in Multipoint Applications, John Goldie, National Semiconductor, Application Note AN-759

[2] Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Computer Society, IEEE-SA Standards Board, IEEE Std 802.3TMTM-2008

[3] Electrical Characteristics of Balanced Voltage Digital Interface Circuits, ANSI/TIA/EIA-422-B-1994, Telecommunications Industry Association, 1994

[4] 다대다 시리얼 통신을 위한 데이터 패킷화, 이진, 한국정보처리학회, 추계학술발표대회 논문집 제16권 제2호

[5] RS-422/RS-485 Communications, Transceivers for EMI-Sensitive Applications, Industrial-Control Local Area Networks

[6] Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems, ANSI/TIA/EIA-485-A-1998, Telecommunications Industry Association, 1998

[7] Jan Axelson, "Serial Port Complete:COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems 2nd Edition", Lakeview Research LLC Madison, WI53704, p.126, 2007

[8] Gong Jian-wei. The Programme Practice of Serial Port Commutation]. Beijing: Electron Industry Book Concer, 2004:315-318