'90 韓國自動制御學術會議論文集 1990. 10.26~27

LLC와 MAC의 구현에 관한 연구

권 욱 현　　안 상 철　　박 정 우　　강 중 용

서울대학교 공과대학 제어계측공학과

A Study on the Implementation of LLC and MAC

WookHyun Kwon　SangChul Ahn　JungWoo Park　JungYong Kang

Dept. of Control & Instrumentation Eng.

Seoul National University

ABSTRACT

In this study LLC and MAC program, which is the base of the communication software, is implemented. This program is tested in the MAP board for IBM PC. The LLC is of class 3, which provides the services of type 1 and 3. The MAC uses the Token Passing Bus Access Method of IEEE 802.4 standard. The LLC is implemented in C language, and MAC in C language and 80186 assembly language. MAC program takes advantage of MC 68824 which is the token bus controller produced by Motorola.

1. 서 론

통신 시스템은 복잡한 소프트웨어 (software)와 하드웨어 (hardware)의 결합체이다. 특히 여기에 들어가는 소프트웨어는 요구되는 통신의 신뢰도에 따라 그 복잡도가 바뀌고, 통신 하드웨어에 따라 그 구성이 바뀌게 된다. 하지만, 국제 표준화 기구 (ISO) 에서는 개방형 시스템 연결 참고 모델 (Open Systems Interconnection reference model)을 발표하고 각 계층의 기능과 통신 규약등을 표준화 시켜서 일관된 소프트웨어의 개발을 가능케 하였다[3]. 본 연구에서는 그림 1 에 나타낸 것과 같은 개방형 시스템 연결 참고 모델 (OSI reference model) 중 계층 2인 데이타 링크 계층 (Data Link Layer)의 구현에 대해 다룬다.

데이타 링크 계층은 통신 시스템 소프트웨어에서 가장 기본이 되는 부분으로 일반적으로 다시 두개의 부계층 (sublayer)로 나뉜다. 이 두 부계층이 바로 LLC (Logical Link Control) 부계층과 MAC (Medium Access Control) 부계층이다. 본 연구에서 구현하는 LLC는 IEEE 표준 802.2의 형태 1과 3 (Type 1,3)의 동작이 가능한 클래스 3 (Class 3) LLC이다. 이는 데이타 링크 계층에서의 논리적 연결 (Logical connection)이 없이 데이타를 전송하게 하는 것으로 일반적인 LLC이다. MAC는 IEEE 802.4 표준에서 정하고 있는 토큰 패싱 버스 억세스 방법 (Token Passing Bus Access Method)을 사용하였다.

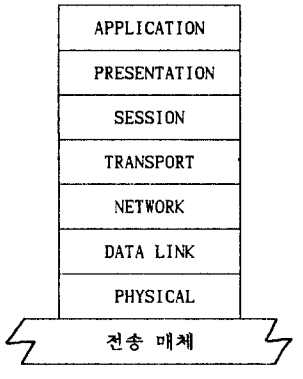

그림 1. 개방형 시스템 연결 참고 모델
Fig 1. Open Systems Interconnection Reference Model

2. LLC와 MAC

LLC는 논리적인 연결을 구성하여 데이타를 전송하게 하는 부계층이다. LLC가 제공하는 서비스 (service)는 모두 3가지가 있다[1,2]. 이들 중 형태 2 (Type 2) 서비스는 연결 중심 서비스 (Connection Oriented Service)로 LLC 수준의 논리적 연결 (Logical Connection)을 이룬 후 데이타를 전송하는 것이다. 이는 신뢰성 있는 데이타 전송을 보장하지만 논리적 연결을 이루는 데 드는 오버헤드 (overhead)가 있기 때문에 상위 계층에서 이미 논리적 연결을 하거나 공장형 네트워크 (network)인 Mini-MAP 같은 경우에 있어서는 사용하지 않는다. 대신 이 때는 논리적 연결을 이루지 않고도 어느정도의 신뢰성을 보장하는 형태 3 서비스를 사용하게 된다. 따라서 본 연구에서는 클래스 3 (Class 3) LLC, 즉 형태 1과 3의 서비스를 제공하는 LLC를 구현하였다. 여기서 제공되는 서비스는 다음과 같다.

형태 1 : 비확인 무연결 서비스 (Unacknowledged Connectionless Service)
- L_DATA.request
  .indication

형태 3 : 확인 무연결 서비스 (Acknowledged Connectionless Service)

- L_DATA_ACK.request
  - .indication
  - .status indication
- L_REPLY.request
  - .indication
  - .status indication
- L_REPLY_UPDATE.request
  - .status indication

MAC는 전송매체 (Medium)의 사용을 제어하는 부계층으로 여기서는 IEEE 802.4 토큰 패싱 버스 억세스 방법을 사용한다. MAC 부계층에서 제공해야하는 서비스는 다음과 같다.

MA_DATA.request
- .indication
- .confirm

MAC가 제공하는 서비스는 이와 같이 단순하지만 이 서비스외에 전송매체 (Medium)를 토큰 버스 방식으로 제어하기 위해서는 아주 복잡한 작업이 필요하다[5]. 따라서 본 연구에서는 이 MAC 부계층을 구현하기위해 토큰 버스 방식을 하드웨어 (hardware)적으로 구현한 MC 68824 토큰 버스 제어기 (Token Bus Controller, TBC)를 사용하였다. 그러면 어려운 전송매체의 관리는 68824 토큰 버스 제어기에게 맡길 수 있고, 대신 우리는 토큰 버스 제어기가 사용하는 버퍼 (buffer)를 관리하고 이 제어기에 알맞은 명령을 내려 전송되고 수신되는 데이타만을 처리하면 된다[6].

3. 프로그램의 구성

전체 프로그램은 LLC와 MAC 부분으로 크게 구분되고 이는 또 다시 송신부와 수신부로 나뉘어져 있다. 프로그램은 C언어와 80186 어셈블리 (Assembly) 언어로 작성하였고 기본적으로 사용되는 자료구조 (data structure)에 관계된 부분은 C언어를, 68824 토큰 버스 제어기 (TBC)에 관계된 프로그램은 어셈블리 언어를 사용하였다.

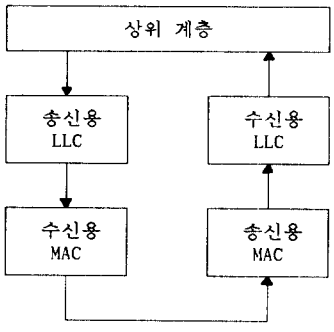

그림 2. 프로그램의 개념적 흐름도
Fig 2. Conceptual Flow Chart of the Program

개념적 모듈 (module)별로 나눈 전체 프로그램의 흐름도는 그림 2 와 같다.

LLC와 MAC는 서로 병렬로 동작하여야 하는데 이를 처리하는 중앙 처리 장치(CPU) 가 순차 처리 밖에 할 수 없을 때에는 그림 2 와 같이 순차적으로 LLC와 MAC를 동작시키되 내부에 상태 표 (state machine)을 두어 병렬로 처리되는 것처럼 한다. 이 때는 프로그램 모듈 (module)을 그림 2 와 같은 순서로 수행하는 것이 빠른 처리 속도를 얻을 수 있다[4].

(1) 큐 구조 (Queue Structure)

전체 프로그램에서 사용되는 가장 기본적인 자료구조 (data structure)는 다음과 같다.

```
typedef unsigned char   uint8;
struct  que
        {
        uint8  ind;  /* indication */
        struct control_block far *cb;
               /*  pointer to control block  */
        }

struct  control_block
        {
        uint8 source;   /* who  sends?   */
        uint8 dest;  /*  who receives?   */
        struct address  loc_ad;
               /*  local address  */
        struct address rem_ad;
               /*  remote address  */
        uint8 token_hold;
              /*  packet received during
                  token hold                */
        uint8 prim_type;  /* primitive type */
        uint8 status;  /* status */
        uint8 prior;  /* priority  */
        uint8 s_class;   /*  service class */
        int   data_len;  /* data length */
        uint8  far *data_ptr
              /* pointer  to data block */
```

이 자료구조는 모듈간 데이타 전송시 사용하는 것으로 데이타에 대한 정보를 저장한다. 모듈간에 실제로 전달되는 것은 control_block structure 인데 struct que에선는 이에 대한 포인터 (pointer)를 가지고 있어서 control_block을 전달할 때 이의 포인터만을 전달한다. 이러한 자료구조는 전체적으로 공통이지만 LLC나 MAC, 또한 상위 계층 사이의 데이타 전송 창구는 나뉘어 있다. 즉, 전송을 위한 자료구조는 struct que의 배열 형태로 구성되어 있고, 이는 전송부와 수신부에 따라 모두 나뉘어 있다. 예를 들면 LLC에서 MAC로 데이타를 보낼 때 사용하는 버퍼 (buffer)는

```
struct  que  mac_llc[TX_QUE_SIZE];
```

이고 MAC에서 LLC로 보낼 때는

```
struct que  llc_mac[RX_QUE_SIZE];
```

를 사용한다. 이처럼 데이타를 보내는 쪽과 받는 쪽을 데이타 전송 창구의 이름으로 하면 데이타를 주고 받을 때 일일이 보내는 모듈과 받는 모듈을 확인할 필요가 없어서 프로그램의 속도를 향상시킬 수 있다. struct que에서 source와 dest 영역은 나중에 IBM PC와 인터페이스

(interface)를 하기위해서만 사용된다. 이러한 자료구조들은 llc_mms, mms_llc, llc_or, or_llc 등이 있다. 여기서 mms, or는 Mini-MAP에서 LLC의 상위 계층으로 MMS나 ORSE를 나타낸다.

이러한 자료구조들은 각각 RX_QUE_SIZE (=10), TX_QUE_SIZE (=5) 개 만큼 씩 잡혀있는데 이에 따라 이들 간의 순서가 필요하게 된다. 즉, 처리를 못하여 데이타가 밀려 있을 때 이들의 처리순서는 마구 정할 수 없고 먼저들어온 서비스 요구 (service request)를 먼저 처리해 주는 FIFO (First In First Out) 구조를 가져야 한다. 본 프로그램에서는 이를 위해 포인터 (pointer)를 사용한 연결 리스트 (linked list) 구조를 사용하는 대신 데이타의 처음 (head)과 끝 (tail)을 나타내는 배열의 인덱스 (index) 만을 변수에 저장하여 FIFO 구조를 가지게 한다. 여기서 데이타는 끝 (tail)에 넣고 처음 (head)에서 꺼내어 쓰게 된다. 실제로 이 FIFO 구조는 링 (ring) 형태로 낮은 인덱스 (0) 에서부터 증가시키면서 사용하고 제일 높은 인덱스 (5 또는 10)의 데이타 버퍼를 사용하면 다시 낮은 인덱스 (0) 에서부터 다시 시작한다. 예를 들면, 이를 위해 사용되는 변수들은 다음과 같다.

```
int   head_mac_llc;
int   tail_mac_llc;
int   head_llc_mac;
int   tail_llc_mac;
```

(2) 상태 변수 (State Variables)

LLC의 형태 3 서비스에서는 상태 변수들이 사용되는데 이들을 저장하는 자료구조 (data structure)는 다음과 같다.

```
struct  tran
        {
        uint8   ind;   /* indication */
        uint8   mac_ad[AD_LEN];
                /* remote MAC address (DA) */
        uint8   lsap;  /* local lsap (ssap) */
        uint8   priority;   /* priority */
        uint8   seq;  /* sequence number V(SI) */
        uint8   timer_on;
                /* TRUE = timer working  */
        int     life_timer;   /* timer value */
        }
struct  tran  tx_state[ST_MACHINE_NUM];

struct  recv
        {
        uint8   ind;    /* indication */
        uint8   mac_ad[AD_LEN];
                /* remote MAC address (SA) */
        uint8   lsap;   /* remote lsap (ssap) */
        uint8   priority;   /* priority */
        uint8   seq;  /* sequence number V(RI) */
        uint8   status;
                /* reception status V(RB) */
        uint8   timer_on;
                /* TRUE = timer working  */
        uint16  life_timer;   /* timer value */
        }
struct  recv  rx_state[ST_MACHINE_NUM];

struct  ack_t
        {
        uint8   ind;   /* indication */
        uint8   mac_ad[AD_LEN];
                /* remote MAC address (SA) */
        uint8   ssap;  /* local lsap (ssap) */
        uint8   dsap;  /* remote lsap (dsap) */
        uint8   priority;   /* priority */
        uint8   s_class;  /* service class */
        uint8   state;  /* LLC state =          */
                        /*  WAIT_A or WAIT_R  */
        uint8   timer_on;
                /* TRUE = timer working */
        uint16  timer;  /* timer value */
        uint16  retry_count;
                /* retry counter */
        uint16  para_len;   /* command length */
        uint8   para[BUFLEN];
                /* old command saved area */
        }
struct  ack_t  ack_timer[ST_MACHINE_NUM];
```

LLC의 형태 3 서비스는 형태 2 서비스에서와는 달리 1 비트 (bit) 순차 번호 매김 (sequence numbering) 방법을 사용한다. 즉, 0과 1을 번갈아 매겨서 데이타를 전송하고 그에 대한 확인 (acknowledge)을 받는다. 위의 struct tran과 struct recv는 송신쪽과 수신쪽의 순차번호 (sequence number)를 보관하는 자료구조이다.

LLC에는 READY, WAIT_A (WAIT_Acknowledge), WAIT_R (WAIT_Response) 의 세가지 상태 (state)가 있는데 struct ack_t는 LLC의 상태를 저장하는 부분이다. 여기서는 상태 뿐만아니라 데이타를 받은 쪽의 확인 (acknowledge)이 없을 때 데이타를 다시보내야 하는데 이를 위해 먼저 보냈던 데이타를 저장하는 역할도 한다.

(3) RWR frame의 처리

LLC 에는 RWNR (Request With No Response), RWR (Request With Response), RESP (RESPonse) 의 세가지 서비스 클래스 (service class)가 있는데 일반적으로 사용되는 것은 RWNR이다. 하지만 때로는 토큰 (Token)을 가지고 있는 스테이션 (station)이 다른 스테이션에 즉시로 응답 (response)을 요구할 경우는 RWR 서비스 클래스를 사용하게 된다. RWR 프레임 (frame)을 보내는 스테이션에서는 RWR 프레임을 보낸후에 응답을 받기 위하여 일정 시간을 기다린다. 따라서 RWR 프레임을 받은 스테이션에서는 이 시간 내에 응답을 만들어 보내주어야 한다. 이를 위해서 프로그램에서는 RWR 프레임을 받으면 68824 TBC에서 186 중앙 처리 장치 (CPU)로 인터럽트 (interrupt)를 걸게 하고, 이를 인터럽트 서비스 루틴 (interrupt service routine)에서 처리하도록 하였다.

RWR 프레임은 형태 3 서비스 에만 적용되고 형태 1 서비스에는 사용되지 않는다. RESP 서비스 클래스는 RWR 프레임의 응답에 사용되는 서비스 클래스이다.

(4) 버퍼 관리

전체 시스템에서 사용하는 버퍼는 실제 데이타가 들어가는 부분과 68824 토큰 버스 제어기에서 사용하는 프레임 디스크립터 (frame descriptor), 버퍼 디스크립터 (buffer descriptor) 등으로 이루어진다. 이것들은 각각 송신용, 수신용으로 나뉘는데 그 구조는 그림 3 과 같

다. 여기서 버퍼의 길이는 각각 1KB 씩 16개가 잡혀있다.

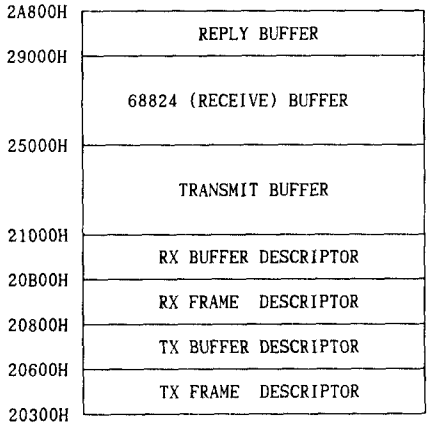

그림 3. 버퍼 메모리 구조
Fig 3. Buffer Memory Structure

프로그램과 68824 TBC는 서로 독립적으로 동작하게 되어 있는데 68824는 스스로 전송 매체를 토큰 버스 억세스 방법으로 관리하고 데이타의 전송과 수신을 담당한다. 68824가 정상적으로 동작하게 하기위해서는 데이타 버퍼를 프로그램에서 관리해서 68824가 사용할 수 있게해 주어야 하는데 프로그램에서는 68824가 사용을 다 한 버퍼를 68824의 빈 버퍼 큐에 넣어주어야 한다.

4. 실험

본 연구에 앞서 연구실에서는 MAP용 인터페이스 보드 (interface board)를 개발하고 여기에 LLC-MAC 프로그램을 메모리에 구워 넣어 테스트해 보았다. 프로그램 개발에 주로 사용한 보드 (board)는 MAP 인터페이스 보드중 IBM PC용 인터페이스 보드로 그 시스템 구성도는 그림 4 와 같다.

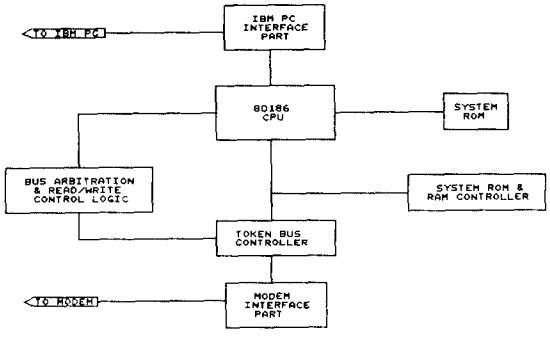

그림 4. IBM PC MAP 보드 시스템 구성도
Fig 4. The Structure of MAP Board for IBM PC

시스템은 인텔 (Intel)의 80186 CPU를 사용하였고 메모리는 램 (RAM) 256 KByte, 롬 (ROM) 128 KByte 이며 시스템 클럭 (clock)은 8 MHz 를 사용하였다. IBM PC 와의 데이타 전송을 위하여 공유 메모리를 두고 이의 사용을 제어 레지스터 (Control Register)를 통해 MAP 보드에서 제어 하도록 하였다. 이 공유 메모리의 사용은 데이타를 가지고 있는 쪽에서 제어 레지스터를 통해 상대방과 협상을 한 후 사용권을 넘겨 받아 사용하도록 되어있다. 프로그램의 테스트는 IBM PC에 LLC의 상위 계층이 존재한다고 가정하고 이 상위 계층의 역할을 해 주는 프로그램을 IBM PC 에서 수행하여 MAP 보드 내부에 있는 LLC-MAC 프로그램을 테스트 하는 방식을 취했다. 개발환경은 그림 5 와 같고 그림에서 IBM PC안에 MAP 보드가 장착되고 IBM PC간의 통신이 이루어지게 된다.

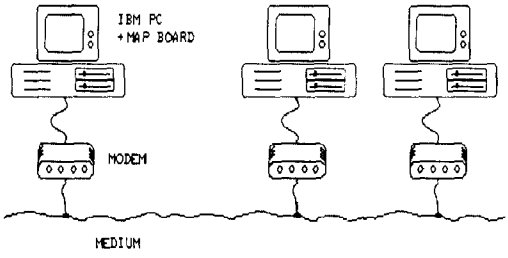

그림 5. 통신 시스템 구성도
Fig 5. The Connection of Communication System

프로그램 개발시에는 일일이 프로그램을 롬 (ROM)에 굽는 것을 피하기 위하여 연구실에서 개발한 모니터 프로그램 (Monitor Program)을 이용, 램 (RAM)에 다운로드 (download)하여 테스트하였고, 개발이 끝난 후에는 프로그램을 롬에 구워 MAP 보드에 장착하였다. 실험에서는 LLC와 MAC 프로그램을 통해서 데이타를 전송할 때 걸리는 시간을 측정해 보았는데 LLC의 형태 1 (Type 1) 서비스를 이용할 때는 1 KB 크기의 패킷 (packet) 당 12 ms 가 걸리고, 형태 3 (Type 3) 서비스를 이용할 때는 23 ms 가 걸렸다. 여기서 형태 3 서비스에서는 확인 패킷 (acknowledge packet)을 받기 때문에 형태 1 서비스보다 시간이 많이 걸림을 알 수 있다. 여기서 측정한 전송 시간은 순수하게 LLC에서 LLC 까지의 전송 시간이 아니고 우리가 가상으로 만든 LLC의 상위 계층으로 부터 다른 스테이션 (station)의 LLC의 상위 계층까지 걸린 시간을 말한다.

5. 결 론

본 연구에서는 통신 시스템 소프트웨어의 기본이 되는 LLC와 MAC를 구현하였다. 이는 Mini-MAP 시스템의 일환으로 개발되어 같이 개발된 IBM PC용 MAP 보드 (board)에서 테스트하였다. 여기서 개발된 LLC는 클래스 3 (Class 3) LLC로 형태 1 과 3의 서비스를 제공하며, MAC는 IEEE 802.4 토큰 패싱 버스 억세스 방법 (Token

Passing Bus Access Method) 을 사용하게 하였다. LLC는 C 언어로 작성되었으며 MAC는 C 언어와 어셈블리 (Assembly) 언어로 작성되어 모토롤라 (Motorola) 의 토큰 버스 제어기 (Token Bus Controller) 인 MC 68824를 제어 하게 하였다.

6. 참고 문헌

[1] IEEE, ANSI/IEEE standard 802.2 Logical Link Control, 1985

[2] IEEE, ISO/IEC JIC 1/SC6/N4960 standards for Local Area Networks : Logical Link Control - Type 3 Operation, Acknowledged Connectionless Service, 1988

[3] Fred Halsall, Data Communications, Computer Networks and OSI, 2nd ed, ADDISON WESLEY, 1988

[4] DeokWoo Kim, "Analysis for the Service Time of User Request on the Mini-MAP Network," Proc. of '89 ISL Winter Workshop, ISL, Seoul National Univ., 1989

[5] IEEE, ANSI/IEEE standard 802.4 Token-Passing Bus Access Method and Physical Layer Specifications, 1985

[6] MC 68824 Token Bus Controller User's Manual, Motorola, 1987