다양한 PRS를 지원하는 분산제어 시스템의 구현에 관한 연구

정회원 김 남 일*

A Study on the Dispersion Control System Supporting Various PRS

Nam - ill Kim* *Regular Member*

요 약

프로토콜 작성기를 이용하여 프로토콜을 운용자가 마음대로 작성할수 있도록 구현하여 PRS를 지원해 보았다. 본 논문에서 구현한 방식으로 프로토콜을 지원할 경우 이미 개발된 PRS는 물론 앞으로 개발 될 PRS까지 지원이 가능하다. 또 프로토콜 개발시간이 드라이브를 이용하는 방식보다 훨씬 단축되며,한번 지원한 프로토콜은 파일로 저장되어 드라이버와 같이 사용할수 있다.

ABSTRACT

This operator supported the PRS so as to make protocol freely by using the protocol maker in case supporting the protocol in the embodied method in the paper. We can support not only the developed PRS already but also the PRS to be developed. The time to develop the protocol can be curtailed by far than the method using the drive and the protocol supported once can be used saving in file like a driver.

I. 서 론

본 논문은 PC에서 다양한 제어기를 운영자 측면에서 프로토콜을 지원하여 원활한 제어가 가능하도록 하는 방법에 대하여 연구한 논문이다. 다양한 제어기를 지원하는 방법으로 '프로토콜 작성기'를 제작하여, 통신을 원하는 제어기의 프로토콜을 운영자가 만들어 통신을 수행하는 방법을 채택했다. 본 논문에서는 다양한 제어기의 지원을 검증하기 위하여 미국에서 알랜 브레드리사에서 제작한 ABPLC의 프로토콜을 지원하여 원활한 통신이 가능한가를 보였다(1). 또한 분산 제어 시스템에서는 다양한 제어기의 지원뿐만 아니라 통신 속도 또한 대단히 중요하다. 따라서 본 논문의 성능 평가에서는 드라이버로 제어기를 지원하는 경우와 프로토콜 작성기에서 제어기를 지원하는 경우의 속도를 비교해 보았다.

II. 분산제어 시스템

분산제어 시스템은 집중제어 시스템이 하나의 시스템에서 집중된 제어를 처리하는 방식에서 탈피하여, 여러 시스템에서 기존의 집중된 제어를 분담하여 처리하고, 그 결과를 중앙의 제어 시스템으로 전달하여 전체공정을 하나의 제어시스템에서 처리하는 것과 같은 결과를 낼 수 있도록 하는 시스템이다.

집중제어 시스템의 주요 단점은 시스템의 일부가 DOWN 되었을 때 전체 제어가 불가능하다는 점과 시스템의 확장 시 전체 시스템을 수정해야만 한다는 점이다. 분산제어 시스템은 이러한 단점을 해결하여 시스템의 일부가 DOWN 되더라도 나머지 시스템 또는 모든 시스템(이중화에 의해서)이 사용 가능하도록 하고, 시스템의 확장 시 그 확장 부분만 수정해도 되도록 한 시스템이다(2).

* 가천길대학 전자통신과, 정회원,
논문번호 : 98001 - 0807, 접수일자 : 1998년 8월 7일

본 논문에서 제시할 분산제어 시스템은 플랜트 공정의 루프(Loop)나 로직 시퀀스(Logic Sequence)를 처리하는 분산된 스테이션(Station)을 PRS(Process Remote Station)라고 정의하고, 이 분산된 스테이션의 결과 값을 보기 위한 그래픽 환경의 중앙 제어시스템을 POS(Process Operating Station)라고 정의한다. 또한 현장에 설치되어 PRS와 직접 통신을 통하여 제어 및 감시를 하는 POS를 서버(Server)라 하고, POS 원거리에 설치하여 서버와 통신을 통해 간접적으로 PRS를 제어 및 감시하는 POS를 클라이언트(client)라고 정의한다.

그림 2-1은 분산 제어 시스템의 전체 구조를 보인 예이다. 그림에서 공용망은 전화 접속망, ISDN, DQDB, FDDI, 인터넷(Internet) 등 기존에 설치되어 있는 망을 가리킨다. PRS는 실제 제어 및 감시를 하는 부분이고 POS는 PRS의 운영자 측면에서 제어 및 감시를 하는 부분이다.

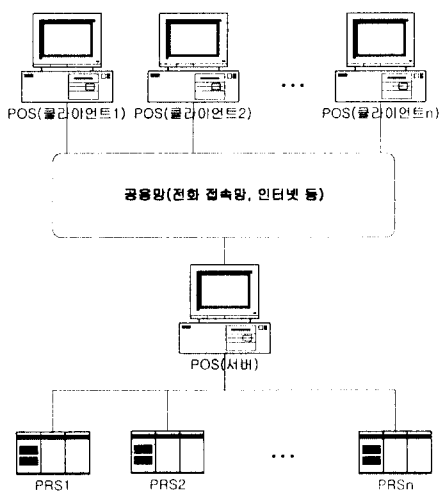

그림 2-1. 시스템 전체 구조

분산제어 시스템의 기능을 크게 4가지로 분류하면 첫째 GUI (Graphic User Interface) 및 데이터 베이스를 이용하여 운영자 측면에서의 감시 및 제어를 하는 POS 기능, 둘째 아날로그 및 디지털 계기를 직접 부착하여 실제 제어 및 감시를 하는 PRS 기능, 셋째 POS와 PRS의 통신을 수행하는 통신기능, 넷째 POS(서버)와 POS(클라이언트)의 통신을 수행하는 망 기능으로 나눌 수 있다(3).

각 기능들은 계층구조를 이루고 상위 계층은 하위 계층의 기능수행에 영향을 받지 않고 구현될 수 있다. 이러한 소프트웨어의 계층 구조는 기능의 추가나 수정이 매우 용이하여 시스템 확장 시 최소의 노력 및 비용이 든다.

III. 다양한 PRS를 지원하는 분산제어 시스템의 구현

1. 프로토콜 작성기 모듈

프로토콜 작성기는 그림 3-1과 같은 구조체를 가지고 프로토콜 파일을 생성한다.

```
typedef struct ProtocolSetup{
    int CommKind;
    BOOL DoubleFlag;
    int DoubleNumber;
    char Description[50];
}PROTOCOLSETUP;

typedef struct SubProtocol {
    char Type;
    char Size;
    int Value;
    char CalcKind;
    char CrcKind;
    BOOL CrcExcept;
    BOOL DoubleExcept;
    char DataKind;
}SUBPROTOCOL;
```

그림 3-1. 프로토콜 구조체

첫째, 전체 설정을 위한 구조로 통신 종류(CommKind) 필드는 LAN 통신, 직렬 통신 그리고 전화 접속 통신 등의 통신 종류를 설정하고, 중복 설정 플래그(DoubleFlag) 필드는 프로토콜 전체에서 전송시 일정한 중복 상수(DoubleNumber)를 중복하여 전송할 것인지를 결정한다. 중복 상수 값이 1일 때 중복 수행을 한다. 중복 상수(DoubleNumber) 필드는 중복 설정 플래그가 1일 때 중복할 상수값을 설정한다. 또한 설명(Description) 필드는 프로토콜 파일에 대한 설명을 입력한다.

둘째 단위 필드의 설정을 위한 구조로 필드 종류(Type) 필드는 필드의 종류를 선택한다. 설정 가능한 종류는 상수, 포트, 주소, 데이터, 계산, 크기, CRC, 주소+크기, AND 마스크, OR 마스크, 비트 위치, 비

트 데이터, 스테이션이 있다. 크기(Size) 필드는 필드의 크기를 설정한다. 상수 (Value) 필드는 필드 종류가 상수일 때 그 상수값을 저장한다. 계산 종류(CalcKind) 필드는 필드 종류가 계산일 때 그 계산 종류를 설정한다. CRC 종류(CrcKind) 필드는 필드 종류가 CRC 일 때 CRC의 종류를 설정한다. CRC 제외(CrcExcept) 필드는 CRC 계산 시 제외될 필요가 있는 필드는 이 값을 1로 설정한다. 중복 제외(DoubleExcept) 필드는 전체 설정에서 중복을 설정했을 경우 필드에서 중복을 수행하지 않을 때 이 값을 1로 설정한다. 데이터 종류(DataKind) 필드는 데이터의 종류를 설정한다. 설정 가능한 데이터 종류는 상수, 문자열, 10진수 문자열 그리고 16진수 문자열 등이 있다.

여러개의 필드들이 모여서 하나의 프레임을 만든다. 프레임은 읽기 요구 프레임, 읽기 프레임, 비트(bit) 쓰기 요구 프레임, 비트 쓰기 확인 프레임, 워드(Word) 쓰기 요구 프레임, 워드 쓰기 확인 프레임, 그리고 요구 확인 프레임 등 7가지가 있다. PRS들은 이 프레임들 중 전체 또는 일부를 요구할 것이다.

2. 검색 작성기 모듈

검색 작성기 모듈은 PRS의 메모리와 POS의 매모리를 매칭(Matching)시키는 부분으로 그림 3-2와 같은 구조를 가지고 파일로 저장된다.

```
typedef struct SubScan{
    char Station[10];
    char PRSPort[10];
    int PRSAddr;
    int POSAddr;
    int Size;
}SUBSCAN;

typedef struct Scan{
    char ProtocolFile[30];
    COMMPORT CommPort;
    SUBSCAN *Data;
    int Size;
}SCAN;
```

그림 3-2. 검색 구조체

첫째 전체 설정을 위한 구조로 프로토콜 파일(ProtocolFile) 필드는 프로토콜 작성기에서 저장한 파일 이름을 저장한다. 이 필드의 이름에 의해 저장된 프로토콜 파일들 중 제어하고자 하는 PRS의 프로토콜을 선택한다. 통신 포트(CommPort) 필드는 통신을 위한 정보를 저장한다. 통신 속도(BoudRate), 포트(Port), 데이터 비트 크기(DataBits), 정지 비트 크기(StopBits), 그리고 패리티 정보(Parity) 등의 멤버(member) 변수가 있다.

둘째, 단위 검색 설정을 위한 구조로 스테이션(Station) 필드는 여러개의 PRS가 연결된 경우 그 PRS들을 구분하기 위하여 사용한다. PRS 포트(PrsPort) 필드는 PRS 포트를, PRS 주소(PrsAddr) 필드는 PRS의 주소를, POS 주소(PosAddr) 필드는 POS의 주소를 그리고 크기(Size) 필드는 메모리 크기를 WORD 단위로 저장한다.

3. 통신 모듈

통신 모듈은 PRS와 실제 통신하는 모듈로 POS에서 PRS로 전송하는 경우와 PRS에서 POS로 전송하는 경우가 있다. 먼저 두 가지 모두의 공통 작업으로 프로그램 실행 시 저장된 프로토콜 파일과 검색 파일을 읽어 메모리에 저장하고 통신 버퍼을 할당한다.

첫째, POS에서 PRS로 전송하는 경우는 먼저 필드 종류(Type)를 검사하여 그 종류에 따라 필요한 데이터를 임시버퍼에 저장한다. 필드 종류가 0이면 상수, 1이면 PRS 포트, 2이면 PRS 주소, 5이면 데이터 크기, 7이면 PRS 주소 + 데이터 크기를 저장한다. 필드 종류가 3이면 전송하고자 하는 데이터를 저장한다. 이 부분은 아날로그 출력시 아날로그 정보를 PRS로 전송할 때 사용한다. 필드 종류가 4이면 계산종류(CalcKind)에 따라 그 계산값을 저장한다. 계산 종류가 0이면 0부터 순차적으로 증가하는 증가값을 저장한다. 필드 종류가 6이면 CRC 종류(CrcKind)에 따라 그 CRC값을 저장한다. CRC 종류가 0이면 CRC 제외(CrcExcept)가 1인 필드를 제외하고 전체 필드를 더한값을 저장하고, CRC 종류가 1이면 [illegible]값을 저장한다. 필드 종류가 8이면 AND 마스크를 저장한다. 이 부분은 디지털 출력시 디지털 정보를 PRS로 전송할 때 사용한다. 출력하고자 하는 비트가 1이면 전체 비트를 1로 마스크하고, 출력하고자 하는 비트가 0이면 그 비트만 0으로 나머지는 1로 마스크 한다. 예를 들어 7번째 위치의 비트를 0으로 설정하고자 한다면 1111111110111111로 이 필드를 채워서 전송한다. 필드 종류가 9이면 OR 마스크를 저장한다. 이 부분 또한 디지털 출력시 디지털 정보를 PRS로 전송할 때 사용한다. 출력하고자 하는 비트가 1이면 그 비트만 1로 나머지는 0으로 마스크하고, 출력하고자 하는 비

트가 0이면 전체 비트를 0으로 마스크 한다. 예를 들어 15번째 위치의 비트를 1로 설정하고자 한다면 0100000000000000로

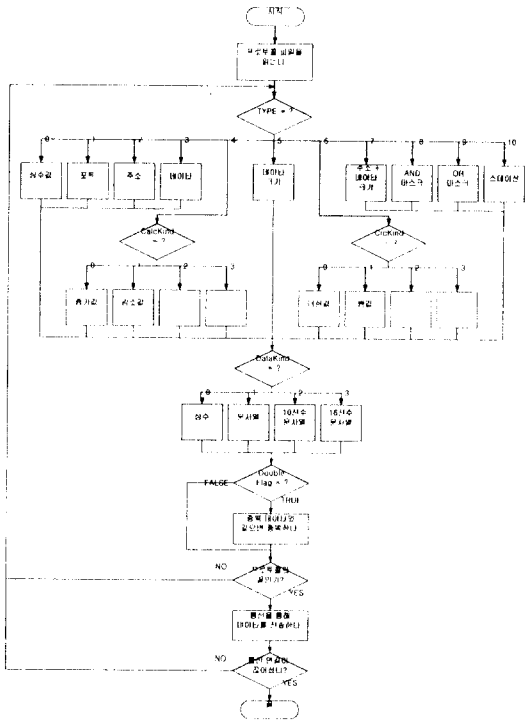

그림 3-3. POS에서 PRS로의 전송 블록도

이 필드를 채워서 전송한다. 필드 종류가 10이면 스테이션(Station)을 저장한다. 필드 종류가 11이면 비트 위치를 저장한다. 디지털 출력시 사용하는 부분으로 태그 정보의 비트 위치를 여기에 저장한다. 필드 종류가 12이면 비트 데이터를 저장한다.

이렇게 저장된 임시 버퍼의 데이터는 데이터 종류(DataKind)에 따라 데이터 값을 변경하여 다시 임시 버퍼에 저장한다. 데이터 종류가 0이면 상수값으로 저장한다. 임시 버퍼의 데이터가 상수이면 그대로, 문자열이면 상수값으로 변경하여 저장한다. 예를 들어 임시 버퍼의 데이터가 상수 9이거나 문자열 "9"(상수 57)이면 상수 9로 변경 저장한다. 데이터 종류가 1이면 문자열로 저장한다. 임시 버퍼의 데이터가 상수이면 문자열로 변경하고, 문자열이면 그대로 저장한다. 예를 들어 임시 버퍼의 데이터가 상수 12이거나 문자열 "12"이면 문자열 "12"로 저장한다. 데이터 종류가 2이면 16진수 문자열로 저장한다. 임시 버퍼의 데이터가 상수이면 16진수 문자열로 변경하고, 문자열이면 상수로 변경한 후 그 상수값을 16진수 문자열로 변경하여 저장한다. 예를 들어 임시 버퍼의 데이터가 상수 12이거나 문자열 "12"이면 문자열 "C"로 저장한다.

중복 설정 플레그(DoubleFlag) 필드가 1이고, 중복 제외(DoubleExcept) 필드가 0면 임시 버퍼의 데이터와 중복 상수(DoubleValue) 필드값을 비교하여 같으면 그 값을 한번 더 삽입하여 통신 버퍼에 저장한다.

통신 버퍼가 필요한 프레임 크기만큼 저장되면 통신을 통하여 PRS로 전송한다. 일련의 작업들을 반복한다.

둘째, PRS에서 POS로 전송하는 경우는 POS에서 PRS로의 전송과 반대 작업을 수행한다. 먼저 들어올 프레임의 크기를 계산하고 그 프레임 크기만큼 데이터를 수신한다. 중복 설정되어 있는 필드의 데이터가 중복되어 들어오면 중복된 데이터를 버린다. 수신된 모든 데이터가 원하는 데이터인지 검사를 하고 맞는 데이터면 그 데이터를 받아들인다.

IV. 실험 및 성능 평가

이 장에서는 다양한 PRS가 지원됨을 보이기 위하여 미국의 알렌 브레드리사에서 제작한 ABPLC를 지원하고, 성능 평가로서 드라이버로 지원한 방식과 본 논문에서 제안하는 방식의 속도 비교를 해 보았다.

1. 실 험

ABPLC는 알랜 브래드리사에서 제작한 PLC로 상당히 복잡한 프로토콜을 가진다. ABPLC 프로토콜은 DLE(상수 16)를 체크하기 위하여 DLE를 제외한 모든 데이터에서 16이 나오면 16을 중복하여 전송한다. 또한 POS에서 전송을 요구하고 PRS에서 데이터를 받기 위하여 요구확인 신호를 주고받아야 한다.

ABPLC를 본 논문에서 구현한 분산제어 시스템에서 지원하기 위하여 프로토콜 전체 설정에서 중복설정을 1로 하고 중복할 상수를 16으로 설정해야만 한다.

1) 읽기 요구 프레임(POS→PRS)

그림 4-1은 ABPLC의 읽기 요구 프레임의 형태를 나타낸 것이다. 그림에서 DLE 필드는 STX와 ETX를 구분하기 위한 부분으로 필드 종류를 상수로 설정하고 그 상수값을 16으로 설정한다. 또한 중복 제외 플레그를 0으로 CRC 제외 플레그를 1로 한다. STX 필

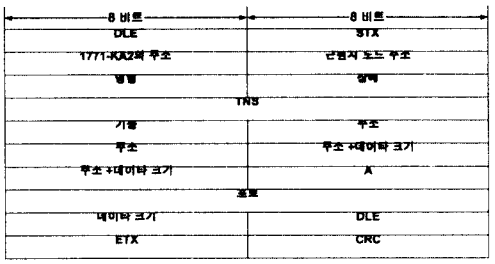

그림 4-1. ABPLC 읽기 요구 프레임

드는 프로토콜의 시작을 나타내는 부분으로 필드 종류를 상수로, 상수값을 2로 CRC 제외 플레그를 1로 설정한다. 1771-KA2의 주소 필드와 근원지 노드 주소 필드는 상수 0으로 설정한다. 명령 필드는 POS에서 PRS로 전송하는 프레임인지 PRS에서 POS로 전송하는 프레임인지를 구분하는 부분으로 POS에서 PRS로 전송하는 경우는 상수 15로 PRS에서 POS로 전송하는 경우는 상수 79로 설정한다. 읽기 요구 프레임은 POS에서 PRS로 전송하는 경우이므로 상수 15로 설정한다. 상태 필드는 상수 0으로 설정한다. TNS(Transaction Check rand number) 필드는 1부터 순차적으로 증가하는 값으로 필드 종류를 계산식으로 계산 형태를 증가로 설정한다. TNS는 크기를 2Byte로 설정해야만 한다. 기능 필드는 기능을 구분하는 부분으로 상수 1로 설정한다. 포트 필드는 1부터 19까지의 메모리 블록 번호로 필드 종류를 포트로 설정한다. PRS의 정보 위치는 포트와 주소로 나타낸다. 주소 필드는 포트가 나타내는 각 블록에서 상대위치를 나타내는 부분으로 필드 종류를 주소로 설정한다. 데이터 크기 필드는 요구하는 데이터의 크기를 워드 단위로 나타내는 부분으로 필드 종류를 크기로 설정한다. 주소+크기 필드는 필드 종류를 주소+크기로 설정한다. A 필드는 상수 6으로 설정한다. ETX 필드는 프로토콜의 끝을 나타내는 부분으로 상수 3으로, CRC 제외를 1로 설정한다. CRC 필드는 에러 체크를 하는 부분으로 필드 종류를 CRC로 CRC 종류를 더한 값으로 설정한다.

2) 요구 확인 프레임(POS↔PRS)

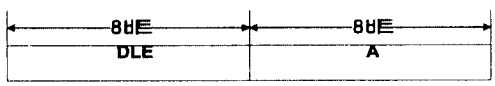

그림 4-2. ABPLC 요구 확인 프레임

그림 4-2는 ABPLC의 요구 확인 프레임의 형태를 나타낸 것이다. 그림에서 A는 상수 6으로 설정한다.

3) 읽기 프레임(POS←PRS)

그림 4-3은 ABPLC의 읽기 프레임의 형태를 나타낸 것이다. 그림에서 명령 필드는 상수 79로 설정한다. 데이터 필드는 실제 PLC의 정보로 이 부분을 POS의 공유 메모리에 읽기 요구 프레임의 데이터 크기 워드만큼 저장한다.

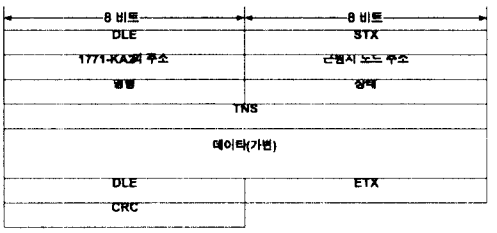

그림 4-3. ABPLC 읽기 프레임

4) 비트 쓰기 요구 프레임(POS→PRS)

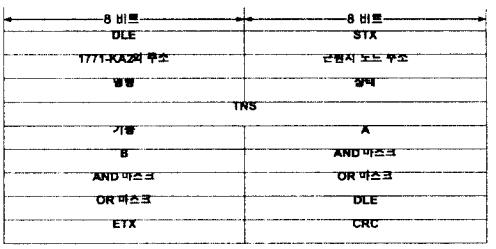

그림 4-4. ABPLC 비트 쓰기 요구 프레임

그림 4-4는 ABPLC의 비트 쓰기 요구 프레임의 형태를 나타낸 것이다. 그림에서 명령 필드는 명령어를 구분하는 부분으로 비트 쓰기 요구 프레임에서는 상수 15로 설정한다. 기능 필드는 기능을 구분하는 부분으로 상수 38로 설정한다. A 필드는 상수 7을 설정한다. B 필드는 상수 0을 설정한다. AND 마스크 필드는 디지털 출력시 사용하는 부분으로 필드 종류를 AND 마스크로, 크기를 2바이트로 설정한다. OR 마스크 필드 또한 디지털 출력시 사용하는 부분으로 필드 종류를 OR 마스크로, 크기를 2바이트로 설정한다.

5) 비트 쓰기 확인 프레임(POS←PRS)

그림 4-5는 ABPLC의 비트 쓰기 확인 프레임의 형태를 나타낸 것이다. 그림에서 명령 필드는 상수 79로 설정한다.

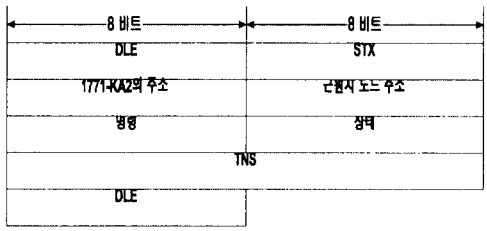

그림 4-5. ABPLC 비트 쓰기 확인 프레임

6) 워드 쓰기 요구 프레임(POS→PRS)

그림 4-6은 ABPLC의 워드 쓰기 요구 프레임의 형태를 나타낸 것이다. 그림에서 명령 필드는 상수 15로 설정한다. 기능 필드는 상수 0으로 설정한다. 주소+크기 필드는 워드 쓰기 요구 프레임에서는 크기가 무조건 1(1워드)이다. 따라서 필드 종류를 주소로 더할 값을 1로 설정한다. A 필드는 상수 6으로 설정한다. 데이터 필드는 전송하고자하는 1워드의 데이터로 아날로그 출력값을 이 필드에 채워서 전송하게 된다.

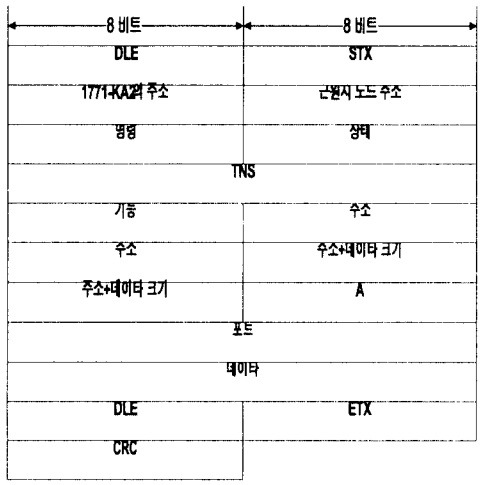

그림 4-6. ABPLC 워드 쓰기 요구 프레임

7) 워드 쓰기 확인 프레임(POS←PRS)

그림 4-7은 ABPLC의 워드 쓰기 확인 프레임의 형태를 나타낸 것이다. 그림에서 워드 쓰기 확인 프레

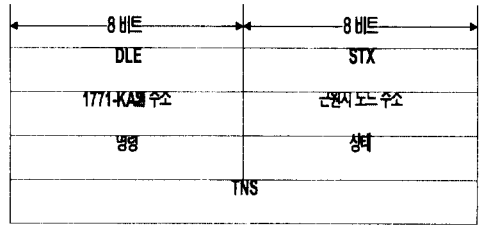

그림 4-7. ABPLC 워드 쓰기 확인 프레임

임은 비트 쓰기 확인 프레임과 마지막에 DLE 필드가 없다는 것을 제외하면 동일하다.

2. 성능 평가

분산제어 시스템에서 다양한 PRS를 지원하기 위하여 본 논문에서 제안하는 프로토콜 작성 방식을 이용할 경우 장점을 보면 다음과 같다. 첫째, 기존에 개발된 PRS는 물론 앞으로 개발될 PRS들까지 지원이 가능하다. 둘째, 분산제어 시스템을 개발한 회사에서 지원하지 않는 PRS도 운영자가 직접 프로토콜을 작성하여 지원할 수 있다. 셋째, 한번 지원한 프로토콜은 파일로 저장되어 드라이버처럼 사용할 수 있어 드라이버로 지원하는 방식의 장점 또한 가진다. 넷째, 프로토콜 개발 시간이 드라이브를 이용하는 방식보다 훨씬 단축된다. 그러나 속도면에서 보면 드라이버로 지원하는 방식은 전용으로 설계되기 때문에 프로토콜 작성 방식보다는 빠를 것이다. 이 속도 차이가 너무 많이 나면 실제 통신을 위해 본 논문에서 제안하는 방식은 사용할 수가 없다. 따라서 이 절에서는 드라이버로 지원하는 방식으로 한솔 컴퓨터에서 개발한 AUTOBASE와 본 논문에서 제안하는 방식과의 통신 속도 차이를 실험하여 본 논문에서 제안하는 방식의 성능을 평가해본다.

실험 방식은 본 논문에서 지원한 PRS중 ABPLC와 PC를 3미터 길이의 직렬통신 선로를 통하여 RS-232-C통신을 수행했고, ABPLC에 부착된 계기 정보의 개수를 증가시키면서 전체 계기의 정보를 한번 읽어 들이는데 걸리는 시간을 비교했다. 표 4-1은 AUTOBASE와 본 논문에서 제안하는 방식의 속도 비교표이다.

표 4-1에서 차이 비율은

차이 비율 = (본 논문 속도 - AUTOBASE 속도) / 본 논문 속도

를 이용하여 구한다.

표 4-1. 통신 속도 비교

(단위 : msec)

개수	AUTOBASE	본 논문	차이 비율(%)
8개(8워드 정보)	55	55	0
16개	110	110	0
32개	220	221	0.0045
64개	441	443	0.0045
128개	881	885	0.0045
256개	1764	1772	0.0045
512개	3530	3546	0.0045

표 4-1에서 보듯이 두 방식의 속도 차이는 거의 0.0045% 정도로 본 논문에서 제안하는 방식이 약간 늦어 졌음을 알 수 있다. 이 속도 차이는 작성한 프로토콜을 이용하여 통신을 수행할 때 드라이버를 이용하는 방식보다는 더 많는 비교 루틴을 수행해야 하기 때문에 발생한다. 그러나 분산처리 시스템에서 0.0045% 만큼의 속도 차이는 제어 및 감시를 하는데 거의 영향을 주지 않는다.

V. 결 론

분산제어 시스템에서 다양한 PRS를 지원하기 위한 가장 좋은 방법은 PRS가 지원하는 프로토콜을 표준화하는 것이고 이에 대한 연구가 진행 되고 있다. 그러나 표준안이 나오더라도 현재까지 생산된 PRS들은 그 표준안에 맞추기가 불가능하다.

따라서 본 논문에서는 '프로토콜 작성기'를 이용하여 프로토콜을 운영자가 마음대로 작성할 수 있도록 구현하여 PRS를 지원해 보았다. 본 논문에서 구현한 방식으로 프로토콜을 지원할 경우 장점을 보면 첫째, 기존에 개발된 PRS는 물론 앞으로 개발될 PRS들까지 지원이 가능하다. 둘째, 분산제어 시스템을 개발한 회사에서 지원하지 않는 PRS도 운영자가 직접 프로토콜을 작성하여 지원할 수 있다. 셋째, 한번 지원한 프로토콜은 파일로 저장되어 드라이버처럼 사용할 수 있어 드라이버로 지원하는 방식의 장점 또한 가진다. 넷째, 프로토콜 개발 시간이 드라이브를 이용하는 방식보다 훨씬 단축된다.

통신 속도 면에서는 드라이버로 지원하는 경우 보다 약 0.0045% 정도 느려진다. 그러나 분산처리 시스템에서 이 만큼의 속도 차이는 제어 및 감시를 하는데 거의 영향을 주지 않을 뿐만 아니라 '프로토콜 작성기'를 이용하는 방식의 장점을 생각하면 충분히 이용가치가 있는 방식이다.

참 고 문 헌

1. Uyles Black, computer Network Protocol Standards and Interface, Perntice-Hall Inc, 1992.
2. Dobrivoje Popovic and Vijay P.Bhatkar, Distributed Computer Control for Industrial Automation, Marcel Dekker Inc, 1990.
3. Simon and Schuster, Data and Computer Communications, Prentice-Hall Inc, 1997.
4. Charles Petzold, Programming Windows 95, Microsoft Press, 1996.
5. David J. Kruglinski, Inside Visual C++, Microsoft Press, 1997.
6. Walter Oney, System Programming for Windows 95, Microsoft Press, 1996.

김 남 일(Nam-ill Kim) 정회원

1954년 6월 24일생

1985년 : 건국대학교 전자공학과 졸업(공학사)

1987년 : 명지대 대학원 전자공학과 졸업(공학석사)

1998년 : 건국대학교 대학원 박사과정 수료

현재 : 가천길대학 전자통신과 조교수

<연구분야> ATM, ISDN, DATA통신