Proceedings of the 11th
KACC, October 1996

생산자동화 시스템에서 필드버스 네트워크 구축 기술 연구

Study on the Implementation of Fieldbus Network in the Manufacturing Automation System

[0]김기암*, 홍승호*, 김지용*, 고성준*

*한양대학교 제어계측공학과 (Tel:0345-400-5665; Fax:406-6639)

Abstract Fieldbus provides real-time data communication among field devices in the process control and manufacturing automation systems. This paper presents an implementation method of Profibus in the experimental model of manufacturing automation system. The manufacturing automation system considered in this paper consists of robots, NC machines, sensors, conveyers and PCs. The application task programs are developed on the basis of FMS/FMA7 communication services which are provided by Profibus application layer. The communication and application programs are developed in the real-time environment.

Keywords 필드버스, 생산 자동화, Profibus, FMS, FMA7, 실시간 프로그래밍

1. 서론

최근 수년간에 걸쳐 스마트 센서, 각종 제어기, NC 머신, 산업용 로봇, PLC 등의 성능이 지속적으로 향상되었고 컴퓨터를 이용한 자동화 시스템의 보급이 증가되었다. 따라서 이러한 자동화 요소들을 네트워크로 연결하여 모든 공정의 통합화를 추구하는 기술의 필요성이 제기되고 있다. 필드버스는 1980년대 후반부터, 생산 현장 필드에 설치된 각종 제어 및 자동화 관련 장비들에서 생성되는 데이터들의 실시간 통신을 지원하는 네트워크 시스템으로, 첨단 생산시스템의 네트워크 구조상 가장 기본이 되는 통신망이다[3,4]. 필드버스에는 IEC/ISA Bus, Profibus, FIP, Fieldbus Foundation 등이 있다. 그러나 국내에 필드버스를 도입한 사례가 드물고 제어 기기 들과 인터페이스가 표준화되지 않아 생산 현장에 이를 구축하기가 어려운 실정이었다.

본 논문은 실제 자동화 장비들을 필드버스 네트워크에 접속시키기 위하여 Parofibus의 응용계층인 FMS를 이용한 네트워크 접속 소프트웨어(network interface S/W)의 구현 방법을 제시한다. 그리고 실시간 다중 작업 프로그램 기법을 사용하여 센서, 컨베이어벨트, 로봇, NC선반, PLC 및 오퍼레이터 스테이션 등을 필드버스에 접속하여 필요한 데이터를 교환하며 통합 운영될 수 있는 첨단 생산자동화 시스템의 실험 모델을 구축한다.

2, 시스템 구성 및 개발 환경

본 연구를 통하여 개발하려는 생산자동화 시스템에서는 필드버스 네트워크로 독일에서 제정(DIN 19 245)된 Profibus를 사용하였다[1,2]. Profibus의 물리계층은 버스 토폴로지를 사용하고 RS485 와 NRZ 코딩방식을 채택하고 한 세그머트에 최대 32개의 노드를 연결 할 수 있다. 데이터링크 계층에서는 논리링크제어를 위해 SDA, SDN, RDR, SDR, CRDR(Cyclic RDR) 서비스를 제공하고 매체접속제어는 여러 개의 마스터 노드를 두어 마스터 노드들 간에는 토큰-패싱과 폴링의 혼합방식으로 동작된다. Profibus 응용 계층은 LLI 와 FMS로 나누어진다. FMS는 사용자에게 다양한 통신서비스를 제공하며 LLI는 이 서비스를 데이터링크 계층에 전달하고 데이터의 흐름을 제어한다.

본 시스템의 구성 도는 그림 1과 같다. 2대의 로봇, 2대의 컨베이어 벨트, NC 선반과, 컨베이어 양단에 부착된 센서 및 오퍼레이터 PC가 Profibus에 연결된다. 가공 대상 품이 컨베이이에 실리게 되면 센서가 물체를 감지하여 컨베이어가 동작되고 일정 지점에 도착되면 로봇이 가공품을 NC 선반으로 옮겨 가공한 다음 다른 컨베이어에 싣게 된다. 물체를 감지한 컨베이어가 구동되고 다른 로봇이 대기한 지점까지 물체를 운반하면 로봇이 물체를 창고에 저장한다. 이들 기기 들을 Profibus에 접속시키기 위하여 MC68302 IMP(Integrated Multiprotocol Processor)에 기반을 둔 IUC와 Smart I/O, PC-보드인 CP5412등의 NIU(Network Interface Unit)가 사용된다. 로봇의 제어기는 RS232를 통하여IUC에 바로 접속이 된다. 센서와 컨베이어를 구동하는 디지털 입력과 출력신호는 Smart I/O에서 제어한다. NC 선반은 PC로 제어하는데 여기에 CP5412보드를 장착하고 오퍼레이터 PC역시 CP5412보드가 내장된다.

이들 NIU들은 Profibus의 물리계층과 데이터링크 계층을 지원한다. CP5412보드인 경우 PC와는 DPRAM(Dual Port RAM)을 통하여 데이터를 교환하며 실제로 이 부분이 어플리케이션 계층과 데이터링크계층의 SAP(Service Access Point)이 된다. IUC인 경우 실시간 커널을 지원하는 OS-9 환경에서 FMS와 FMA7 및 사용자 프로그램을 개발하였다.

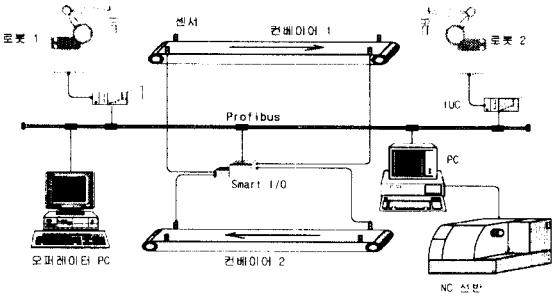

그림 1. 생산 자동화 시스템의 실험 모델

3. FMS의 기능과 구현

FMS(Fieldbus Message Specification)는 Profibus의 어플리케이션 계층(DIN 19 245 Part 2)에 해당한다. FMS는 MAP의 MMS에 대응되며 통신에 사용되는 객체, 서비스, 통신모델 등을 정의하며 사용자에게 실질적인 통신 서비스를 제공한다. 산업현장에서 사용되는 자동화기기와 응용 프로그램들이 필드버스에 접속되기 위해서는 여러 기기 들간에 상호 호환성과, 독립성 등이 보장되어야 한다. 그러므로 FMS를 구현하기 위해서는 객체지향적인 방법이 사용되어야 한다. FMS는 다음과 같은 기능을 제공한다.

- VFD 관리
- OD(객체사전) 관리
- Context(연결) 관리
- Variable(변수) 관리

- Event(사건) 관리
- Domain(영역) 관리
- Program Invocation(프로그램 기동) 관리
- Access Protection(접근) 관리

3.1 VFD 관리

VFD(Virtual Field Device), 즉 가상 기기 모델은 FMS의 중요한 부분 중 하나이다. VFD는 FMS가 실제 생산 장비의 자원과 기능을 추상화된 객체로 정의한다. FMS에서 각 객체는 속성(attribute)에 의해 구별되는데 VFD의 속성은 다음과 같이 정의된다.

```
typedef struct VFD{
unsigned long        vfd_number;
char        vendor_name[MAX_VFD_STRING_LENGTH];
char        model_name[MAX_VFD_STRING_LENGTH];
char        revision[MAX_VFD_STRING_LENGTH];
char        profile_number[2];
unsigned char        logical_status;
unsigned char        physical_status;
}
```

VFD 관리에는 Create, Status, Unsolicited-Status, Identify와 같은 서비스들이 지원되며 하나의 VFD는 각기 하나의 OD(Object Dictionary)를 갖는다. 그러므로 CRL(Communication Relationship List)의 *vfd_pointer* 파라미터는 앞에서 정의된 vfd_number 파라미터와 동일하여야 한다.

3.2 OD 관리

FMS에서는 통신 객체들을 정의하기 위해서 OD(Object Dictionary)를 사용한다. OD의 내부 구성요소로는 OD 전체 구조에 대한 정보를 가진 Object description object dictionary가 있다. 시스템 초기화 시 정의되며 구동 도중에 변경될 수 없는 정적인 OD로서는 Static type dictionary(ST-OD) 와 Static object dictionary(S-OD)가 있다. 전자는 데이터 형 또는 데이터 구조가 정의되고, 후자는 통신에 사용할 간단한 변수, 배열, 레코드, 도메인, 이벤트에 대한 객체가 명시된다. 또한, 시스템 구동 도중에 변경되는 동적인 OD로서 Dynamic Variable list dictionary(DV-OD)와 Dynamic program invocation dictionary(DP-OD)가 있다. 전자는 프로그램 구동 중 생성되거나 소멸되는 변수들의 목록을 포함하며, 후자는 여러 개의 도메인으로 구성되는 프로그램 객체를 다루고 있다. OD는 FMS가 하위 계층에 넘겨줄 SDU(Service Data Unit)을 생성하는데 있어서 실질적인 데이터가 되므로 매우 중요하다고 하겠다. 특히 ST-OD는 사용자가 직접 데이터형을 정의할 수 있어야 하며, 동적인 OD를 생성하기 위해서는 동정 메모리 할당 기법을 사용하여야 한다. OD의 구조는 다음과 같고 각 개체는 id로 구별된다.

```
typedef    struct OBJECT_DESCR{
 union{
   OD_OBJ_DESCR_HDR     od_obj_descr;
   OD_ST_DT_OBJECT      dt_obj_descr;
   OD_ST_DS_OBJECT      ds_obj_descr;
   SIMPLE_VAR_OBJECT    s_var_obj_descr;
   ARRAY_OBJECT         a_var_obj_descr;
   RECORD_OBJECT        r_var_obj_descr;
   VAR_LIST_OBJECT      vlist_obj_descr;
   DOM_OBJECT           dom_obj_descr;
   EVENT_OBJECT         eve_obj_descr;
   PI_OBJECT            pi_obj_descr;
 }id;
}T_OBJECT_DESCR;
```

OD는 또한 구동 중에 갱신되거나 사용자가 이에 대한 정보를 얻을 수 있어야 하므로 Get-OD, Initiate-Put-OD, Put-OD와 같은 서비스를 제공해야 한다. 그런데 OD를 변경시키면 통신 상대방의 OD에 영향을 미치게 되므로 주의를 기울여야 한다. 그러므로 각 노드에서는 Source-OD와 Remote-OD를 동시에 관리하게 된다. Remote-OD는 통신 상대의 OD를 복사해서 상대방의 OD가 변경될 때 마다 갱신하게 된다.

3.3 Context(연결) 관리

FMS의 서비스는 Connection Oriented 방식과 Connectionless 방식으로 동작된다. Connection-Oriented 방식은 Confirmed-Service-Primitive를 사용하여 서비스를 수행한다. 그러므로 송신단과 수신단간에 통신 관계를 설정하고 해제할 필요가 있는데 이것이 Context 관리이다. 이 서비스로는 통신관계를 설정하는 Initiate, 해제하는 Abort, 관계설정을 거부하는 Reject서비스가 있다.

3.4 Varialbe(변수), Domain(영역), PI(프로그램기동) 관리

통신 관계가 설정된 후 OD에서 정의된 변수, 영역, 프로그램 등의 객체를 클라이언트 노드에서 요구하거나 전송하면 서버에서 해당 객체의 정보를 주거나 갱신하게 된다. 변수는 배열, 레코드 등을 말하며 도메인은 전송 프레임의 최대 길이를 넘지 않도록 여러개의 세그먼트를 분리하여 전송한다.

다음은 배열의 데이터구조이다. 다른 객체는 지면관계상 생략한다.

```
typedef struct ARRAY_OBJECT{
 unsigned int        index;
 unsigned char       obj_code;     // object code
 unsigned char       length;  // length of an element in octets
 unsigned int        index_of_type; // logical address of type
 unsigned char       nof_element; // number of array elem.
 T_ACCESS            access;       // access right structure
 unsigned long       local_address; // local address
 unsigned char*      name[MAX_OBJECT_NAME_LENGTH];
 unsigned char       extension[MAX_EXTENSION_LENGHT];
}
```

3.5 Event(사건)관리

필드버스에서 사건은 중요한 의미를 갖는다. 각 기기 들에서 생성되는 알람 신호와 같은 중요한 정보를 다중 전송(Multicast 또는 Broadcast)방식으로 여러 노드에 알리게 된다. 사건 객체는 단순한 변수가 될 수 있고 배열, 레코드, 혹은 사용자가 정의한 데이터형이 될 수도 있다.

3.6 Access protection(접근) 관리

사용자는 응용프로그램에서 FMS의 OD를 이용하여 데이터를 전송하거나 수신한다. 이 때 응용프로그램과 FMS를 연결할 통로가 필요하다. FMS는 Access protection기법을 이용하여 이 통로를 관리하게 된다. 각 OD에 논리적 또는 물리적 어드레스를 부여하거나 이름을 사용하게되며 여러 개의 다수의 응용프로그램인 경우 id를 부여해 선택적 접근을 하도록 한다.

4. FMA7의 기능과 구현

FMA7(Fieldbus Management Layer7)는 버스 시스템 전체의 구성과 통신관련 기능의 관리 역할을 수행한다. FMA7 서비스의 요청은 내부(local)와 원격(remote)에서 모두 가능하다. 원격서비스는 통신 상대방에게 FMA7서비스를 요청하는 것으로써 상대방은 자신의 내부서비스를 이용하여 이를 실행하고 결과를 통보해 주어야 한다. FMA7은 다음과 같은 기능을 제공한다.

- Context-Management(연결관리)
- Configuration-Management(시스템 구성 관리)
- Fault-Management (오류처리 관리)

4.1 Context-Management(연결관리)

원격서비스를 실행하기 위해서 상대방과 통신관계를 설정하고 해제한다. 이는 FMS의 Context 설정과 동일하다. 특히 시스템 구성과 관리에 관계되는 기기 들을 접속하기 위해서 기본 연결 관리(default management connection)를 정의하는데 FMA7 원격서비스의 표준 응답자의 역할을 하게 된다.

4.2 Configuration-Management(시스템 구성 관리)

버스 시스템을 구성하기 위해서는, 통신 관계 목록(Communication Relationship List)이나 SAP(Service Access Point), 또는 데이터 링크계층의 변수들에 대한 관리가 필요하다. CRL은 각 통신 기기 들간 통신 관계의 순차적인 목록인데 FMS는 이를 참조하여 데이터 프레임을 생성할 때 주소, SAP, 프레임 길이, 서비스 종류 등을 결정한다. 그러므로 이는 시스템 구성 시 매우 중요한 역할을 하게 된다. CRL에서 통신 관계는 CR(Communication Reference)에 의해 순차적으로 기입된다. 즉 하나의 통신 연결은 이에 대응하는 CR을 갖게 된다. CRL은 헤드와 엔트리로 구성되는데 헤드는 CR값이 0이다. 엔트리의 데이터 구조는 다음과 같다.

```
typedef struct CRL_STATIC{
unsigned char  loc_lsap;     //내부 SAP
unsigned char  rem_add;      //통신 상대 주소
unsigned char  rem_segm;     //통신 상대 세그먼트 주소
unsigned char  rem_lsap;     // 통신 상대 SAP
unsigned char  conn_type;    //통신 연결 종류(마스터-슬레이브)
unsigned char  lli_sap;      // FMS 또는 FMA7
unsigned char  multiplier;   // 주기적 연결에서 폴링 주기 결정
unsigned char  conn_attr;    //연결 특성(D, I, O)
unsigned char  max_scc;      // send confirmed counter
unsigned char  max_rcc;      // receive confirmed counter
unsigned char  max_sac;      // send acknowledged counter
unsigned char  max_rac;      // receive acknowledged counter
unsigned long  ci;           // 연결 제어 시간 간격
unsigned char max_pdu_snd_high;
    //상위 우선 순위 전송 FMS-PDU의 최대 길이
unsigned char max_pdu_snd_low;
   // 하위 우선 순위 전송 FMS-PDU의 최대 길이
unsigned char  max_pdu_rcv_high;  //수신 FMS-PDU의 길이
unsigned char   max_pdu_rcv_low;
unsigned char  feature_supp[FEAT_SUPP_LEN];
                    //지원되는 FMS서비스
char*      symbol[MAX_CRL_SYMBOL_LENGTH];
                    //CRL Symbol의 최대 길이
unsigned long   vfd_pointer; // VFD에 대한 포인터
unsignec  char  extension[MAX_CRL_EXTENSION_LENGTH];
}
```

FMA7은 Load-CRL-Loc, Read-CRL-Loc, Load-CRL-Rem, Read-CRL-Rem 등의 서비스를 통하여 내부 또는 원격 스테이션의 CRL을 기입하거나 읽을 수 있다. 그리고 위의 데이터 구조에서 알 수 있듯이 사용자가 모든 값을 결정하기에는 너무 번거롭기 때문에 CRL과 OD를 편리하게 사용할 수 있는 도구가 필요하다.

앞에서도 언급하였듯이 LLI와 FDL사이에서 데이터의 전송이 실지로 일어나는 곳이 SAP(LSAP, Link Service Access Point)이다. FMA7 서비스는 사용자에게 이들 SAP의 동작여부를 알려주는데 LSAP-Status-Loc, LSAP-Status-Rem 서비스를 사용한다. 버스 시스템의 구성 및 성능을 결정하는 스테이션 주소와 전송속도, 각종 타이머 값은 FMA1/2계층에서 관리하는데 FMA7를 이용하여 접근할 수 있다. 시스템 초기화 시 이 들 버스파라미터는 사용자가 결정할 수 있어야 하며 구동 중에도 일부 변경할 수 있다. 지원되는 서비스는 Set-Bus-Parameters, Set-Value-Loc, Set-Statistic-Counters, Read-Value-Loc등이 있다.

4.3 Fault-Management(오류 처리 관리)

오류가 발생하였을 때 각 계층을 초기화 할 필요가 있다. FMA7-Reset 서비스는 각 계층의 통신 소프트웨어를 모두 초기화한 다음 재 동작한다. LLI와 FDL에서 사건(Event)이 발생하였을 때 이를 사용자에게 알려야 한다. FMA7-Event서비스는 사건이 발생한 계층과 이유를 어플리케이션 또는 사용자에게 알려준다.

5. PC에서 FMS와 응용 프로그램 개발

본 논문에서 생산자동화 시스템의 NIU로써 PC에 CP5412보드를 장착하여 FMS와 응용프로그램을 개발하였다. CP5412보드는 물리계층과 데이터링크계층의 기능을 제공하는데 프로세서로 NEC V25+가 사용되고 PC와의 인터페이스를 위해 64K Dual-Port-RAM을 사용하고 이의 주소는 PC에서 프로그램할 수 있다.

필드버스의 응용프로그램은 실시간 다중작업의 요건을 만족하여야 한다. 특히 수행해야 할 작업이 많고 통신 관계가 복잡한 경우 한 스테이션에서 동시에 처리해야 할 통신 서비스가 많아지게 된다. DOS는 실시간 OS의 기능을 제공하지 못하므로 실시간 다중작업 프로그램 기법을 사용하였다. 각각의 작업은 프로세서의 시간을 분할하여 사용하게되는데 스케쥴러가 다음에 수행될 작업을 결정한다. 스케쥴러는 일정 시간마다 현재 작업을 큐에 저장하고 우선 순위가 가장 높은 작업을 시작한다. 수행되는 작업들을 동기화시키고 작업들간에 데이터를 전달하기 위해서 Flag, Pipe, Mailbox등을 사용한다.

기본적으로 FMS와 응용프로그램의 하위계층과의 데이터교환은 Request, Indication, Response, Confirmation 등의 서비스 프리미티브를 이용한 전송/수신 인터페이스 구조를 갖는다. 각 데이터는 Service-Description-Block과 Data-Block으로 구성된다. 전자는 제공되는 서비스의 특성을 결정하고 후자는 그 서비스가 전송하는 데이터를 정의한다. Service-Description-Block의 데이터 구조는 다음과 같다.

```
typdef struct T_PROFI_SERVICE_DESCR{
unsigned int     comm_ref;   //communication reference
unsigned char   layer;       // FMS, FMA7, User
unsigned char   service;     // service_id
unsigned char primitive;
    //primitive(request, indication, response, confirmation)
unsigned char   invoke_id;   // task ID
unsigned int     result;     // positive or negative result
}T_PROFI_SERVICE_DESCR;
```

Data-Block은 각 서비스가 제공하는 데이터의 구조를 가지며 PROFIBUS DIN 19245 part 2에 정의되어 있다. FMS와 응용프로그램의 전체 구조는 그림 2와 같다.

기본 작업으로는 FMS&FMA7, receive_cnf_ind, transmit_req_res이 있고 어플리케이션 작업은 필요한 만큼 추가할 수 있다. receive_cnf_ind와 transmit_req_res작업은 하위계층과 데이터를 교환한다. receive_cnf_ind는 하위 계층의 confirm 또는 indication 프리미티브를 수신하여 큐에 저장한다.

transmit_req_res는 FMS와 FMA7 또는 어플리케이션에서 생성된 request, response 프리미티브를 하위계층에 넘겨준다. FMS&FMA7은 하위계층에서 생성된 데이터를 각 어플케이션에

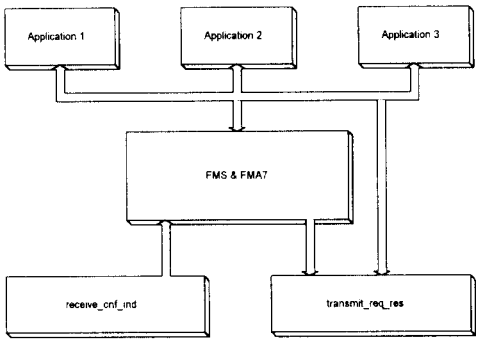

그림 2. PC에서 FMS와 응용프로그램 구조

할당하고 에러나 사건이 발생한 경우 이를 처리하고 어플리케이션이나 하위계층에 알린다. 각 어플리케이션 작업에서 생성된 데이터는 mailbox를 사용하여 FMS&FMA7이나 transmit_req_res에게 전달한다.

6. OS-9에서 FMS와 응용 프로그램의 개발

본 논문의 생산자동화 시스템의 또 다른 NIU로써 IUC를 사용하였다. IUC는 MC68302프로세서를 사용하는데 5개의 타이머와 6개의 시리얼 포트를 지원한다. 특히 통신을 위해 Risc 프로세서를 사용하는데 dual-port RAM을 통해 Profibus 프로토콜을 지원한다. 그러므로 프로세서가 다른 어플리케이션에 훨씬 더 많은 시간을 소비할 수 있다.

OS로는 실시간 커널을 제공하는 OS-9이 사용된다. 여러 개의 메모리 모듈로 구성되며 프로세서, 메모리, I/O, 파일 등에 관계되는 시스템 서비스들을 실시간 커널에서 제공한다. 각 프로세서에 대한 시간을 할당하기 위하여, 우선 순위를 갖는 round-robin 방식의 스케쥴러가 사용되며, 각 프로세서들의 동기화와 내부 통신을 위해 Data Modules, Pipe, Event, Alarm, Signal등이 사용된다.

FMS와 응용프로그램의 하위계층과의 데이터 교환은 PC에서와 마찬가지로 서비스 프리미티브의 전송/수신 인터페이스 구조를 갖는다. 데이터 역시 Service-Description-Block과 Data-Block으로 구성된다.

응용프로그램의 전체 구조는 그림 2에 나타난 바와 같이 PC에서와 동일한 구조를 갖는다. 다중 작업 처리를 위해 signal handler를 이용하여 signal 값에 따라 작업전환을 하게 된다. 또한 어플리케이션으로서 시리얼 통신 기능을 제공해야하는데 이는 SCF(Sequential Character File Manager)를 이용하여 사용자가 개발한 SCF-type device driver에 I/O 서비스를 요구함으로써 구현할 수 있다.

FMS 서비스를 시작하기 위해서는 VFD를 정의 생성하고 버스파라미터, OD, CRL을 작성하여 LLI에 로드해야하며 VFD의 상태를 설정하고 수신 버퍼를 정의해야 한다. 이들은 실제 생산자동화 시스템의 성능을 결정하는 요인으로써 매우 중요하다.

버스파라미터의 전송속도, 스테이션 지연시간, GAP update factor 값의 변화에 따라 데이터 전송에 영향을 미친다. 또한 OD 작성 시 불필요한 전송 객체는 삭제시켜야 하며 CRL에서 통신 관계는 최대한 단순화해야 한다. 한 스테이션에서 여러 개의 통신 관계가 설정되었을 때 이를 처리하는 문제가 발생하는데 Service-Description-Block의 Invoke_ID를 사용하여 각 서비스를 구분해야 한다.

7. 응용 프로그램 S/W 구조

이제까지 FMS와 FMA7의 개발환경과 구현에 대하여 언급하였다. 본 논문에서는 이를 이용하여 그림 1의 생산자동화 시스템을 구축하였다. 오퍼레이터 PC는 Robot와 NC에 기준 좌표 값과 데이터를 전송하고 각 스테이션들의 상태를 모니터하며 사용자가 전체 시스템을 제어하도록 한다. 오퍼레이터 PC에서 구현된 FMS서비스로는 변수관리, Domain관리, Event관리 등이 있다. Smart I/O에서는 각 센서와 컨베이어의 구동상태를 Broadcast(write) 서비스를 이용하여 주기적으로 모든 스테이션에 알린다. Robot는 Smart I/O에서 수신된 센서를 통하여 로봇을 구동하고 이를 오퍼레이터 PC와 NC에게 알린다. NC는 센서와 로봇 구동신호를 이용하여 NC를 구동한다. 본 논문에서는 지면관계상 오퍼레이터 PC의 인터페이스 S/W 구조에 대해서만 기술하기로 하며, 이에대한 구조가 그림 3에 주어져 있다.

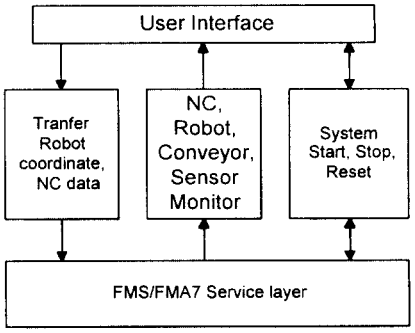

그림 3. 오퍼레이터 PC 작업구조

CP5412보드가 내장된 PC가 오퍼레이터 PC로 사용되며 전체 프로그램은 실시간 다중작업 기법을 사용하였다. 오퍼레이터 인터페이스 S/W는 사용자 인터페이스 태스크(Operator), 로봇 기준 좌표와 NC 구동 데이터 처리 태스크(Robot coordinate, NC data), 각 스테이션의 상태를 모니터하는 태스크(NC, Robot, Conveyor, Sensor Monitor), 전체 시스템의 시작, 종료, 초기화 처리 태스크(System Start, Stop, Reset), 통신을 담당하는 FMS/FMA7 서비스 태스크(FMS/FMA7 Service layer)로 구성된다. 다른 응용프로그램들 역시 실시간 프로그래밍 환경에서 개발되었으며 자세한 사항은 지면 관계상 생략한다.

8. 결론 및 추후연구사항

본 논문에서는 필드버스를 이용하여 생산자동화 시스템을 구축하는 경우에, 실제 자동화 장비들을 네트워크에 접속하는데 필요한 FMS와 응용프로그램의 요구사항과 구현방법을 제시하였다. 추후 연구사항으로는 이 생산자동화 모델을 이용하여 각 스테이션에서 발생한 메시지의 전송 지연 시간을 측정하고 버스파라미터와 시스템의 성능과의 관계를 비교한다. 또한 Profibus 시뮬레이션 모델의 결과와 실제 시스템의 차이점을 분석하여 시뮬레이션의 정확성을 향상시키기 위한 방안을 제시한다.

참고문헌

[1] DIN 19 245 Profibus Standard Part 1: 1991
[2] DIN 19 245 Profibus Standard Part 2: 1991
[3] S. H. Hong, "Scheduling Algorithm of Data Sampling Times in the Integarated Communication and Control Systems," *IEEE Trans. on Control Systems Technology*, Vol. 3, No. 2, June 1995
[4] 홍승호, 김기암, 김지용, 고성준, "분산제어 및 자동화 시스템과 필드버스," 제어・자동화・시스템공학회지, 제2권 제4호, 1996년 7월