'93 한국자동제어학술회의논문집 (1993.10.20~22)

고기능 로봇통신

전재욱, 김진기, 김형철, 최상진°, 송철호, 김성진, 김동일, 김성권

Implementation of High Performance Communication for Industrial Robot Controllers

Jae Wook Jeon, Jin-Ki Kim, Hyeong-Cheol Kim, Sang-Jin Choi
Chul-Ho Song, Sung-Jin Kim, Dong-Il Kim, Sungkwun Kim
Control R & D Team FA Research Institute Production Engineering Center
SAMSUNG Electronics

Abstract

A communication system in an industrial robot controller is proposed for the implementation of an efficient FA system. Different from existing industrial robot controllers which use binary I/O for the communication with external systems, the robot controllers having the proposed communication system can exchange various information with external systems. Furthermore, the robot motion and the communication with external systems can be performed simultaneously. The structure of hardware and software in the communication system is explained to show how to acheive these two operations simultaneously.

1. 서 론

산업용 로봇의 응용은 초기의 단순한 작업에서부터 점차적으로 복잡한 작업으로 그 범위를 넓혀나가고 있다. 즉 초기의 로봇시스템은 고정된 외부 환경에서 예정된 작업을 수행하는 정도의 기능만을 보유하였으나, 최근에 들어서는 변화하는 환경에 대처하여 작업을 수행할 수 있는 기능도 점차적으로 필요하게 되었다. 이러한 기능은 각종 상품의 고기능화와 다양한 소비자의 욕구로 인하여 더욱 필요하게 되었다. 이러한 기능을 보유하기 위해서는 로보트시스템과 주위 각종 외부시스템과의 원활한 정보 교환과 더불어 이 정보를 로봇의 동작에 쉽게 적용할 수 있도록 하여야 한다. 현재 산업계에서는 로보트시스템과 외부 시스템과의 정보교환을 위해 로보트제어기가 보유한 이진값을 갖는 I/O를 주로 사용한다. 이러한 I/O를 이용하여 자동화라인을 구현하면 다양한 정보교환이 어렵게 되고 많은 라인 연결이 필요하게 되어 효율적인 라인을 구성하기가 어렵다. 따라서 좀 더 효율적 자동화라인 구성을 위해 로보트시스템과 외부시스템과의 다양한 정보교환이 필요하다. 이것은 여러대의 로보트시스템을 하나의 주컴퓨터에 연결하여 전체 라인을 제어할 때 더욱 필요로 하게 된다.

본 논문에서는 이를 위하여 외부시스템과 고속통신이 가능하며 이 통신을 통한 정보로써 로보트의 동작을 여러가지로 교정할 수 있는 기능을 위해 별도의 CPU로 구성된 통신시스템을 보유하는 로보트시스템을 구현하였다. 이 통신시스템은 통신기능만을 수행하는 별도의 CPU를 갖고 있기 때문에 로보트가 동작중에도 외부와의 통신이 이루어질 수 있으며 다양한 정보교환 또한 가능하도록 구성되어 있다.

우선 기존의 외부시스템과의 정보교환방법과 제안된 통신시스템을 이용한 외부시스템과의 정보교환방법을 간단히 비교한 후, 제안된 통신시스템의 하드웨어와 소프트웨어의 구조에 관해 설명할 것이다. 그리고 이 통신시스템을 이용하여 통신기능을 수행했을 때 걸리는 시간을 실제로 측정하여 외부시스템과 통신을 하는 데 지연시간이 발생하지 않는다는 것을 보여 줄 것이다.

2. 제안된 통신시스템의 필요성

기존 산업계의 자동화라인처럼 로보트제어기에서 이진값을 갖는 I/O를 이용하여 외부시스템과 정보를 교환할 때는 간단한 정보의 교환만 가능하게 된다. 이러한 I/O를 이용하여 PLC에서 로보트를 제어하고자 할 때는 로보트의 프로그램의 실행과 중지, 비상정지, 서보드라이버의 전원공급 또는 차단및 원점복귀와 같은 비교적 단순한 기능만을 수행할 수 있게된다. 이는 이진값의 I/O로서는 쌍방향 통신 및 명령에 대한 feedback을 구현하기가 어렵기 때문이다. 그러므로 제어기의 각종 parameter의 변경, 프로그램의 작성 및 backup, 위치 포인터의 작성, 로보트의 현재 상태, 프로그램의 실행 상태 등의 정보는 얻을 수 없게된다. 따라서 이러한 단순 기능의 통신으로서는 효율적인 자동화라인의 구성이 어렵게 된다. 특히 이러한 문제는 여러대의 로보트를 PLC를 이용하여 자동화라인을 구성하고 전체 라인의 정보관리를 하나의 컴퓨터에서 하고자 할 때 더욱 커지게 되는 것이다.

그러므로 효율적 자동화라인 구성을 위해 외부시스템과 다양한 정보를 빠른 시간내에 교환할 수 있는 기능을 보유한 통신시스템이 필요하게 된다. 이와 같은 필요성에 따라 본 논문에서는 실시간 통신이 가능한 통신시스템을 보유하

는 로보트제어기를 개발하였다. 이 로보트제어기는 실시간 고속 통신을 위해 별도의 CPU로 구성된 통신전용시스템이 보유하고 있다. 이 통신시스템은 다양한 정보교환이 가능하며, 외부와의 정보교환 방법은 RS-232C 방식을 채택 하였다. 외부기기에서 RS232C를 통해서 들어온 데이터는 DP-RAM을 통하여 로보트시스템의 main CPU로 전달되어지며 그 내용에 따라 로보트의 동작을 결정할 수 있다. 또한 로보트시스템에서 외부기기로 전달하고자 하는 내용은 main CPU에서 DP-RAM을 통해서 통신시스템에 전달되며, 통신시스템은 이 데이터를 RS232C를 통해서 외부기기로 보내게 되는 것이다.

3. 실시간 고속통신을 위한 H/W

고성능 로보트 제어기의 통신시스템은 그림 1에 나타낸 것과 같이 intel 8086을 CPU로 하여, 로보트시스템의 main CPU와는 DP-RAM을 통해서 통신을 하며 외부와는 82510 직렬통신칩을 이용한 RS-232C 직렬통신을 하고 있다. main CPU와의 통신을 DP-RAM을 이용함에 따라 데이터 전송은 고속처리가 가능하며, 이에 따라 main CPU가 직접 외부 RS-232C 통신을 할 경우보다 훨씬 적은 load가 걸리게 된다. 이렇게 통신 전용 CPU를 두게 됨으로서 main CPU는 사용자 프로그램 수행 및 로보트 제어를 더욱 효과적이고 정밀하게 할 수 있게 된다.

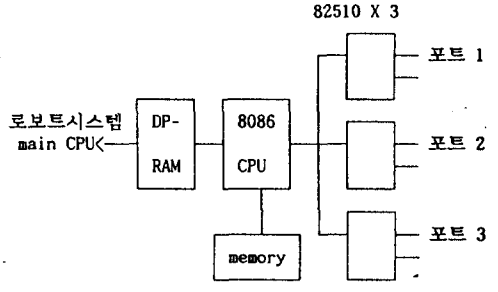

그림 1 통신시스템의 블럭 다이아그램

통신 시스템의 사양은 표 1과 같으며, 64KB RAM은 RS-232C 각 포트당 20KB의 버퍼와 4KB의 시스템 memory로 할당되어 있다.

표 1. 통신 시스템 사양

CPU	intel 8086-2 8MHz
ROM	64 KB
RAM	64 KB
DP-RAM (IDT7130)	1 KB
RS-232C Controller	82510 3개
Interrupt controller	8259

통신시스템의 DP-RAM은 각 포트별로 340 Bytes(data 339 byte, flag 1 byte)씩 할당하였고, 각 RS-232C 포트는 할당된 DP-RAM의 영역만을 사용하므로 포트간 데이터 충돌을 방지 하였다. 그림 2는 DP-RAM의 구성도이며, pflag는 각 포트에 할당된 DP-RAM의 사용자를 표시하며, int2rbt는 통신시스템에서 로보트시스템의 main CPU로 인터럽트를 요청하기 위한 것이고 int_ack는 DP-RAM을 통한 main CPU의 인터럽트 요청에 대한 인터럽트 acknowledge 신호용이다.

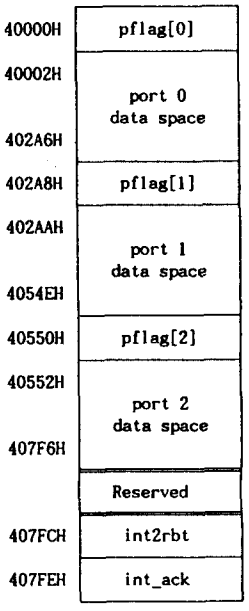

그림 2 DP-RAM의 구성도

4. 통신 알고리듬 및 프로토콜

통신은 8086에 인터럽트를 요청하므로서 시작되며 인터럽트는 3개의 RS-232C 포트와 DP-RAM에서 요청할 수 있다. 인터럽트가 4개의 장치에서 동시에 발생하는 경우를 위해 우선 순위를 RS-232C 포트 0, 1, 2 그리고 DP-RAM순으로 하였으며, 우선 순위는 순환을 하도록 하였다. 통신 프로그램은 크게 주 처리 루틴과 인터럽트 처리 루틴으로 나눌 수 있으며, 데이터 수신은 인터럽트 처리 루틴내에서 수신을 끝내도록 하였다.

RS-232C 사양은 표 2와 같으며 인터럽트를 이용하여 실시간 처리가 가능하도록 하였다.

표 2 RS-232 사양

인터페이스	RS-232C
전송속도	9600 BPS
동기방식	비동기방식
전송코드	8 data, 1 stop, even parity

RS-232C를 통한 외부와의 데이터 입출력은 그림 3과 같은 프로토콜을 가지고 통신을 하게 된다.

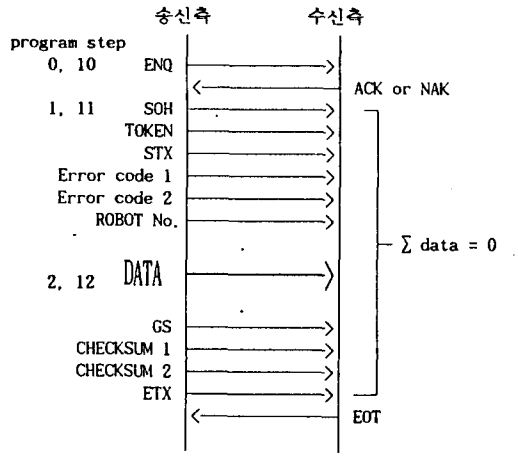

그림 3 통신 프로토콜

그림 3에서처럼 RS-232C를 통한 외부와의 통신은 통신을 요청하는 송신측의 ENQ(Enquiry)에 의해 시작되며, 수신측은 송신측의 ENQ에 대하여 수신측과의 통신이 가능한지 여부를 판단하여 통신 가능(ACK를 송신) 혹은 불가능(NAK를 송신)을 통신 요청측에 알려준다.

송신측의 ENQ에 따라 통신이 가능하게 되면 송신측은 수신측으로 다음과 같은 순서로 정보를 보내게 된다. 먼저 송신의 시작을 표시하는 SOH을 전송하고, 처리해야하는 일을 나타내는 token 그리고 송신 데이터의 시작을 나타내는 STX, 송수신의 error 및 그 처리 과정에서의 error를 나타내기 위한 BCH(Binary Coded Hexadecimal) code, 송신 데이터, 송신 에러 여부를 검사하기 위한 checksum, 송신 데이터의 끝을 나타내는 ETX를 보내게 된다.

수신측은 송신측의 ENQ에 따른 ACK를 전송한 뒤 SOH에 의해 데이터의 시작과 ETX에 의해 끝을 인식하여 정상적인 완료를 나타내는 EOT를 통신 요청측인 송신측에 알려준다. 또한 통신중에 통신 선로의 이상, 하드웨어상의 고장등에 따르는 통신 송수신측의 문제점을 해결하기 위하여, 통신 포트를 리셋 시키도록 하는 기능도 있다.

통신 시스템에서 RS 232C를 통하는 송수신은 송신시에는 그림 4, 수신시에는 그림 5와 같은 알고리듬이다. 그림 6은 DP-RAM의 인터럽트 요청에 따른 인터럽트 처리 루틴 알고리듬이다.

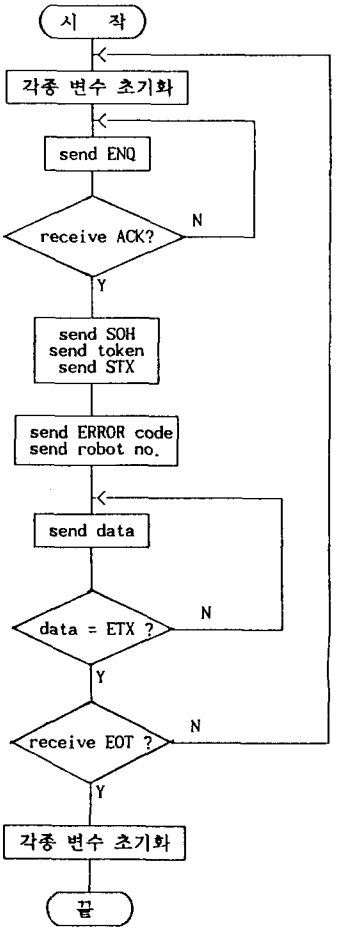

그림 4 송신측의 흐름도

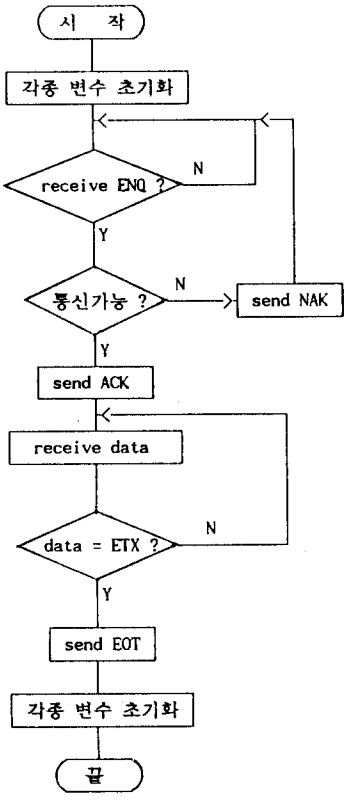

그림 5 수신측의 흐름도

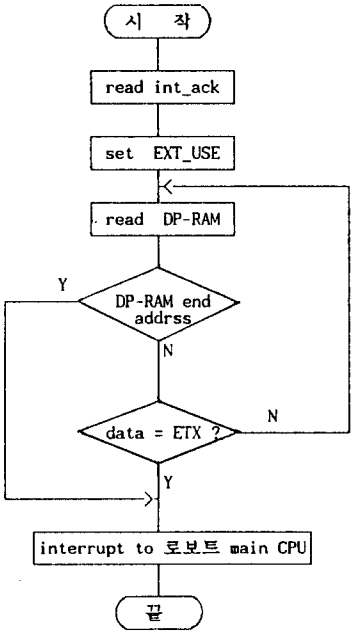

그림 6 DP-RAM 인터럽트 처리 흐름도

4-1. 주처리 루틴

주 처리 루틴에서는 주변장치에서 인터럽트가 가능하도록 인터럽트를 enable 시켜주고, 또한 차례대로 각 포트의 통신 상태를 검사하여 그 처리를 한다.

만약 RS-232C 수신 데이터의 양이 340 Byte 이상이 되면 DP-RAM으로 송신을 하고, DP-RAM에서 읽은 데이터는 RS-232C로 송신을 한다. 아래는 통신 주 처리 루틴의 흐름이다. step은 그림 3의 program step과 같다.

```
main_routine()
{
   initialize_8259;
   initialize_variable;
   interrupt_enable;

   while(1){
      for(port = 0; port < NO_PORT; port++){
         switch (step[port]){
            case  1 : if(received data size >= 340)
                        write_data_to_dpram(port);
                      break;
            case  2 : write_data_to_dpram(port);
                      break;
            case 10 : send_rs232(port, ENQ);
                      step[port]++;   break;
            case 12 : send_data_to_port(port);
                      break;
            default : break;
         }
      }
   }
}
```

4-2. 인터럽트 서비스 루틴

인터럽트는 DP-RAM, RS-232 포트 0, 1, 2에서 발생하며, 이 모두는 로보트 main CPU나 외부에서 통신 시스템으로 데이터를 전송할 경우에 발생한다. 인터럽트 루틴에서는 데이터를 읽어 들여서 buffer에 저장하여 주 처리 루틴에서 목적지로 전송하도록 한다. 인터럽트의 처리 시간은 실시간 통신의 가장 중요한 요소이며, 가장 긴 인터럽트 처리 시간은 인터럽트 요청 평균 주기보다 짧아야 실시간 통신이 가능하게 된다.

rs232_interrupt_service는 RS-232로부터 데이터 수신시의 인터럽트처리알고리듬이며, DP_RAM_interrupt_service는 DP-RAM으로부터의 데이터 수신시 발생하는 인터럽트의 처리 알고리듬이다.(그림 6 참고)

```
rs232_interrupt_service()
{
   receive data from rs232;

   switch (step) {
      case  0 : if(data == ENQ && pflag == 0){
                   set EXT_USE flag;
                   send_rs232(ACK);
                   step++;
                }
                else  send_rs232(NAK);
                break;
      case  1 : if (data == ETX)  step++;
                save data to buffer;
                break;
      case 11 : if (data == ACK)  step++;
                else  step = 10;
                break;
      case 13 : if (data == EOT)  step = 0;
                break;
      default : break;
   }
}

/*-------------------------------------------------*/

DP-RAM_interrupt_service()
{
   port = int2ext & 0x0F;   /* INT, acknowledgement */
   set EXT_USE flag;        /* EXT is using DP-RAM */

   if(step == 0 || step == 12) {
      dpr_ptr = dpr_start_address;
      if(step == 0)  step = 10;

      for( ; dpr_ptr <= dpr_end_address; dpr_ptr++) {
         buffer = *dpr_ptr;
         if(*dpr_ptr == ETX)  return;
      }

      interrupt request to Robot;
   }
}
```

5. 프로그램 결과 및 수행 시간

통신 시스템의 프로그램은 C와 assembly로 구성되어 있으며 앞에서 설명된 알고리듬으로 프로그램하였다. 통신 프로그램의 속도와 정확한 수행에 가장 중요한 인터럽트 처리 시간은 아래 표 3과 같으며, 시간 측정은 HP 8086 emulator를 이용하였다.

표 3 인터럽트 처리 시간

RS-232 interrupt	82 μsec
DP-RAM (1 byte)	52.553 μsec

RS-232C의 9600bps일 때 1 byte 입력 주기는

$$9600 \text{ bps} / 11 \text{ bit} = 1146 \ \mu\text{sec}$$

이며, 3개의 포트에서 동시에 발생하는 경우를 고려한 RS-232 평균 입력 주기는

$$1146 / 3 = 382 \ \mu\text{sec}$$

가 된다.

그러므로 RS-232C에서 데이터를 입력 받아 DP-RAM으로 보내기까지의 필요한 시간은

$$\text{RS-232C 처리 시간} + \text{DP-RAM 처리시간} = 82 + 52.553 = 134.553 \ \mu\text{sec}$$

이며, 이것은 3개의 포트 RS-232C의 평균 입력 주기인 382 μsec 보다 훨씬 작으므로, 통신 시스템은 3개의 RS-232C 포트를 통해 들어오는 데이터를 DP-RAM을 통하여 로보트의 main CPU로 전달할 수 있으며, 3개의 포트에 대한 실시간 통신이 가능 하다.

6. 결 론

본 논문에서는 기존의 자동화라인을 구성할 때 생기는 여러가지 제약점을 해결하기 위하여 통신시스템을 제안하여 로보트시스템에 구현하였다. 이 통신시스템은 외부기기와 다양하고 빠른 통신이 가능하게 하여 효율적인 자동화라인 구성이 가능하게 한다. 이 논문에서 개발된 통신시스템은 실제 라인에 적용하여 미비점을 보완할 것이며, 향후 보다 우수한 기능의 통신시스템 구성을 위해서는 통신 알고리듬 및 프로토콜과 통신중 error에 대한 연구가 있어야 할 것이다.

참 고 문 헌

1. 삼성전자, "고성능 다목적 로보트 제어기 사용자 매뉴얼", 1993

2. Intel, "8086 User's manual"

3. HP, "HP 8086 emulator manual"