2004년도 대한전기학회 하계학술대회 논문집 2004.7.14-16

PCI 방식의 HSIO(High Speed I/O)시스템의 개발

조규상*, 이종운**
동양대학교 컴퓨터학부*, IT전자공학부**

Development of HSIO(High Speed I/O) System with PCI Interface

Gyu Sang Cho*, Jong Woon Lee**
School of Computer Sci.&Eng.*, School of IT & Electronics Eng.**, Dongyang Univ.

Abstract - In this study, a system that has a high speed digital data I/O and distributive structure is developed and the hardware and software of the system are described in detail. PCI master card to PC slot has maximum 63 slaves which are connected by Ethernet cables and can handle 16 I/O points. The system has some features : easy expansion by adding slaves as needed, space and wiring advantage with distributed characteristics, and select from a range of slave devices that fits best for the use.

1. 서 론

PLC(Programmable Logic Controller)는 산업용 기계들이나 정밀기계 등의 주제어기로 산업현장에서 일반적으로 많이 사용되고 있다. PC가 고가이던 시절에 PLC는 프로그래밍 언어보다 쉬우며 직관적으로 사용할 수 있으며 편리하다는 장점을 갖고 있다. 그러나 최근에 PC의 급속한 발전으로 인해 안정성과 고성능이 확보되면서 산업현장에서 PC가 주 제어기로 사용되는 PC-based Control 시스템이 많이 사용되고 있다.

PC-based 제어 시스템을 산업현장에 적용함으로써 기존의 PLC/DCS보다 실시간 제어성, 개발의 편리성, 신뢰성, 확장성 등 여러 우수함이 있어서 투자비용 절감, 유지보수 비용절감, 엔지니어링 기술을 확충하는 등의 장점이 있다는 사례가 발표되었다[1].

PC를 주제어기로 사용하는 경우는 I/O카드에 장치를 연결하여 사용하게 된다. 그러나 좁은 PC의 슬롯에 많은 I/O 카드를 장착해야 하는 어려움이 있다. 이런 단점들을 극복하기 위한 방법으로 USB 방식을 사용한 연구 사례가 있다[2]. 이것은 PC에 표준 장착되어, 통합 운영이 가능하고, 케이블링의 문제점을 개선한 것이다. 이 구조는 USB 포트당 4개의 슬레이브를 갖기 때문에 최대 64 포인트까지만 사용할 수 있다. 그러므로 포인트 수가 많아지면 더 많은 USB 포트 연결이 필요하기 때문에 CPU의 부담이 가중되는 구조이다. 후에 이런 구조적인 단점을 개선하기 위하여 매스터와 슬레이브간의 통신 속도가 매우 빠른 구조를 가지며 Ethernet 케이블을 사용하여 간단하게 모든 마스터와 슬레이브들을 연결하여 분산형으로 배치가 가능한 입출력 제어시스템을 개발하였다[3]. 이 방식도 USB를 사용하여 PC와 통신하는 구조를 갖는다. USB방식은 일반 사용자들이 사용하기에 매우 편리하다는 장점이 있지만 산업용으로는 사용하는 경우에는 케이블의 착탈시마다 디바이스를 다시 설정하는 불편함이 존재하고 속도가 비교적 느리다는 단점을 갖고 있다.

본 연구에서는 매우 빠르게 동작하고, 정해진 주기마다 접근시간의 변동없이 안정적으로 동작하며, 분산배치가 가능하고, 케이블 절감효과가 있는 시스템을 설계하기로 한다. 2장의 본론에서는 전체 시스템의 개요에 대한 언급을 한 후에, 전체 시스템의 제어방식, 설계한 PCI 마스터 카드, 슬레이브 장치의 설계방법 그리고 본 연구에서 제작한 소프트웨어에 대해서 알아보기로 한다. 3장에서 결론을 맺기로 한다.

2. 본 론

2.1 전체 시스템의 개요

전체 시스템은 PC의 슬롯에 PCI 방식으로 인터페이스가 설계된 PCI 마스터 카드와 마스터 카드에 RJ45 커넥터와 10BaseT형식의 케이블로 연결된 슬레이브 장치로 구성된다. 이 방식은 슬레이브에서 슬레이브로 연결될 수 있는 구조를 갖고 있어 분산 배치가 가능하다. 각 슬레이브들은 16포인트의 입력/출력 단자를 가지며 여기에 각종 입출력 센서, 모터, 램프 등의 제어대상 기기들이 연결된다. 한개의 PCI 마스터 카드당 최대 63개의 슬레이브를 연결할 수 있고, 각 슬레이브는 16포인트를 가지므로 PCI 마스터 카드당 포인터수는 63x16=1008개의 포인터를 갖는다.

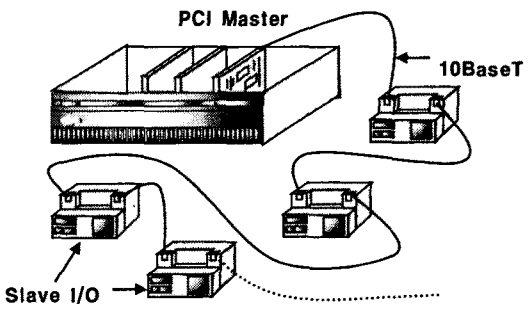

그림 1 전체 시스템 구성도

2.2 시스템의 제어방식

이 시스템은 마스터 장치와 슬레이브 장치 간의 통신을 위한 신호를 고속으로 전송한다. 본 연구에서의 신호

의 방식은 Half/Full-duplex가 가능하다. 본 연구에서는 Full-duplex방식의 마스터 장치는 자동 폴링방식으로 최대 63개의 슬레이브와 통신을 한다. 통신 방식은 하드 로직으로 이루어지므로 매우 빠르고 안정된 통신을 할 수 있다. 통신속도는 3M/6M/12Mbps중에서 선택 가능하다.

마스터 장치의 메모리에는 슬레이브 명령을 저장하는 command, 슬레이브로 부터의 입력을 저장하는 메모리(D_i) 슬레이브에 출력하는 내용이 저장되는 출력메모리(D_o)가 할당되어 있다. 한 슬레이브 당 각 메모리의 크기는 2바이트 씩 사용된다. 이 이외에 카운터 영역과 데이터 영역이 할당되어 있으나 본 연구에서는 이 메모리 영역을 사용하지 않는다.

63개의 슬레이브를 제어하기 위한 command 메모리 영역은 002h에서 07Eh까지 사용된다. 입력과 출력 데이터 저장을 위한 입력 메모리(D_i)와 출력 메모리(D_o)는 각각 082h-0FEh, 102h-17Eh까지 사용된다.

각 슬레이브에 해당 명령은 해당하는 2바이트 공간에 기록하도록 되어 있다. 명령의 상태값에 따라 입출력 동작이 매 스캔 주기마다 이루어진다. command 메모리 중에서 최하위의 000h는 특수하게 사용된다. 매 스캔 주기마다 스캔할 슬레이브의 범위를 설정하는데 사용된다. 이 메모리 안에 기록된 값이 0Fh라면 최대 가능한 63개의 슬레이브 범위를 모두 스캔하지 않고 1번 슬레이브에서 0Fh번 슬레이브까지만 스캔하므로 사용하지 않는 슬레이브에 대해서 스캔하지 않도록 범위를 줄이는 데 사용된다.

슬레이브에서의 입출력 동작은 슬레이브의 타입에 의해서 결정된다. 만일 입력이 가능한 슬레이브인 경우라면 입력된 값들이 D_i에 기록하면 입력으로 작용하고, 출력이 가능한 경우라면 D_o에 기록을 하면 슬레이브로 출력되는 방식으로 동작한다. 어떤 슬레이브로 데이터를 출력하는데 사용하는 출력 메모리 D_o는 command의 상태와는 무관하게 독립적으로 동작하여 언제나 슬레이브에 전송하는 것이 가능하다. D_i 데이터는 command의 상태가 0이거나 8일 때 매 스캔 주기마다 슬레이브로부터 갱신된 입력이 저장된다. 이 영역은 슬레이브로부터의 입력을 위해 사용되는 곳이기 때문에 동작중일 때 이 영역에 대한 쓰기 동작은 불가능하다.

2.3 PCI 마스터 카드의 설계

산업용 I/O는 PC 호스트로부터 디바이스 I/O까지 통신을 하는데 있어 호스트의 부담을 줄여줄 수 있는 시스템설계가 필요하다. PCI버스에서는 병렬I/O를 이용하는 경우와 시리얼통신을 이용하는 분산 I/O등의 방식이 가능하다. 본 연구에서는 분산방식으로 입출력 구조를 가지면서 호스트의 부담을 줄여 고속으로 입출력이 가능한 시스템을 설계하기로 한다. 즉, 호스트 입장에서는 병렬로 입출력을 하는 것 같지만 실제로는 직렬통신 방식을 사용하여 각각의 슬레이브에 데이터가 전달되고, 이렇게 전달 받은 데이터를 슬레이브에서 다시 입출력하는 방식으로 분산 입출력을 구현한다. 그림 2는 본 연구에서 설계한 마스터 카드의 방식을 나타내고 있다. 이것은 직렬 I/O방식을 사용하여 분산 I/O가 가능하며, 직렬-병렬 변환기능과 통신을 관장하는 전용칩을 사용하여 호스트가 슬레이브의 I/O를 직접관장하지 않고 마치 메모리를 다루듯이 동작하는 방식을 사용하여 호스트의 통신 부담을 줄이는 장점이 있다.

PCI 버스 I/F에는 PLX technology사의 PLX9030 칩을 사용하였고, 직렬-병렬 변환과 통신을 관장하는 마스터 I/O칩으로는 StepTechnica[4]사의 MKY33을 사용하였다. 여기에 버퍼 메모리를 추가하고, 시리얼 통신부에 절연 트랜스포머를 추가하여 전이중(full-duplex) 통신방식으로 H/W를 설계 구현하였다.

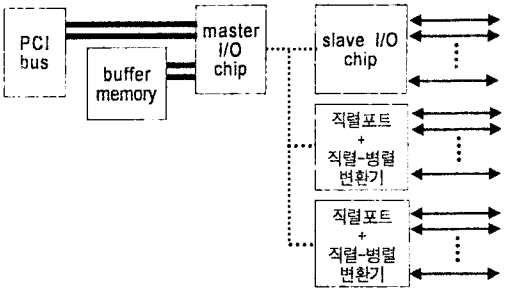

그림 2 PCI 마스터카드의 구성도

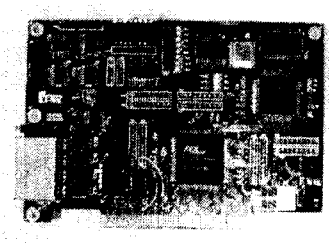

그림 3 제작된 PCI 마스터 카드

통신속도는 6Mbps로 고정하여, 입력 8포인트+출력 8포인트, 입력 16포인트, 출력 16포인트 등의 슬레이브 모듈을 연결시, 각 모듈 당, 30.3μscc 정도의 스캔시간이 걸리고, 현실적으로 8개의 슬레이브(총 128포인트)를 연결할 때는 0.24 mscc, 32개의 슬레이브(총 512포인트) 연결시에도 1mscc이내에 전 I/O를 업데이트할 수가 있다.

2.4 슬레이브 장치의 설계

슬레이브 장치는 입력과 출력의 포인트 수에 따라서 3종으로 나뉜다. 타입 I은 입력과 출력의 포인트 수가 각각 8개인 경우이고, 타입 II는 입력의 포인트 수가 16개이고 출력 포인트 수는 없는 경우이다. 타입 III은 입력의 포인트 수는 없고, 출력 포인트만 16개인 경우이다.

각 슬레이브에 사용되는 MKY34는 데이터의 입력을 위해 사용되는 핀이 16개 구성되어 있고, 출력을 위해 사용되는 핀이 역시 16개 구성되어 있다. 모든 슬레이브는 똑 같은 구성으로 되어 있기 때문에 타입이 다르게 설계되어야 하는 경우는 슬레이브 들을 서로 구분하기 위해서는 그림 4,5,6에서와 같이 입·출력 핀을 연결하는 방법에 따라 서로를 구분하는 방법이 사용된다.

그림 4에서 입력부의 상위 바이트가 항상 슬레이브의 타입 정보를 갖고 있다. 그중 상위 니블은 입출력 각각 8포인트의 타입의 경우 0x60의 값을 가지며, 하위 니블은 출력부의 상위 바이트 중 하위 니블과 연결되어 있다. 즉, 타입을 인식하기 위하여 출력부 상위바이트에 특정값(예로 0x05)을 쓴 후 입력부의 상위값을 읽었을 경우 읽혀진 데이터의 값은 0x65가 되는 것을 이용하여 타입 I이라고 알아낸다.

그림 6에서 출력 16점의 경우 상위 니블은 0xA0의 값을 가지며, 하위 니블은 출력부의 상위 바이트중 하위 니블과 연결되어 있다. 즉, 타입을 인식하기 위하여 출력부 상위바이트에 특정값(예로 0x05)을 쓴후 입력부의 상위값을 읽었을 경우 읽혀진 데이터의 값은 0xA5가 된다. 이런 특징을 이용하면 타입 II를 인식할 수 있다.

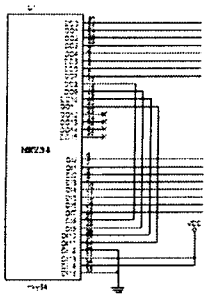

그림 4. 슬레이브 타입 I

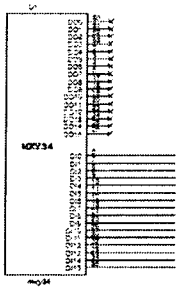

그림 5 슬레이브 타입 II

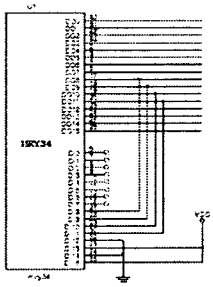

그림 6. 슬레이브 타입 III

표 2 슬레이브 타입구분

타입	입력 포인트 수	출력 포인트 수
I	8	8
II	16	0
III	0	16

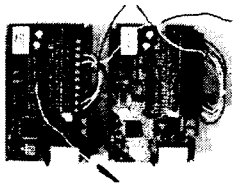

그림 7 슬레이브 장치 타입I

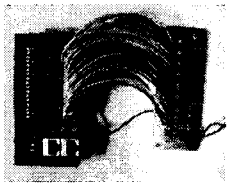

그림 8 슬레이브 장치 타입III

그림 5의 입력 16점의 경우는 특정한 출력에 대한 입력 패턴 방법을 적용할 수 없으므로 앞의 두 경우가 아닌 경우 이 슬레이브는 타입 II로 인식한다. 그림 7,8은 앞에서 설명한 방식대로 제작된 슬레이브의 사진이다.

2.5 소프트웨어의 구성과 기능

본 연구에서의 소프트웨어는 용도에 따라 3종류로 제작된다. PC의 PCI 슬롯에 장착된 PCI 마스터 카드와의 통신을 위해서 디바이스 드라이버 프로그램이 작성되고, 사용자의 응용 프로그램에서 사용할 수 있는 어플리케이션 라이브러리가 제작된다. 또한, 마스터와 슬레이브간의 통신을 위해서 하드웨어에 내장되는 펌웨어가 작성된다.

디바이스 드라이버는 Microsoft의 Windows XP나 2000에서 사용될 수 있도록 제작되었다. 디바이스 드라이버 프로그램은 Windows NT의 드라이버 방식에 기반을 두고 있고 있는 WDM (Windows Driver Model)[5]을 사용한다. 이 방식은 PnP기능, Power Management, WMI기능이 추가 되었고, Class driver를 지원하고 있다. 본 연구의 PCI 용 디바이스 드라이버 프로그래밍은 Windows XP/2000용의 DDK (Driver Development Kit)을 사용하여 Visual C++ 6.0 IDE에서 작성되었다.

본 연구에서 개발한 라이브러리들은 PCI의 시작과 종료 시에 필요한 설정을 하기위해 사용되는 InitializePCI(), FinishPCI() 함수가 사용된다. 슬레이브의 상태 검사는 ClearAllMemory() GetSlaveScanRange(), SetSlaveScanRange(), GetSlaveType(), CheckAllSlave()등의 함수가 사용된다. 각 슬레이브 장치에 연결된 포인트들에서의 입력과 출력은 ReadInputWord(), WriteOutputWord(), ReadInputByte(), WriteOutputByte(), ReadInputBit(), WriteOutputBit()이 사용된다. 이 함수들의 기능설명은 표 3에 정리되어 있다.

표 3 라이브러리 함수명과 기능설명

구분	함수명	기능설명
시작, 종료	InitializePCI()	PCI카드의 초기화를 위한 함수
	FinishPCI()	PCI 카드의 사용을 종료함
슬레이브 상태 검사	ClearAllMemory()	마스터 카드의 메모리에 기록된 모든 내용을 클리어하는 기능
	GetSlaveScanRange()	현재 설정되어 있는 슬레이브 스캔 범위의 값을 구하는 기능
	SetSlaveScanRange()	슬레이브 스캔범위의 값을 새로운 값으로 설정하는 기능
	GetSlaveType()	마스터에 연결된 슬레이브 장치의 타입을 정수값으로 구하는 기능
	CheckAllSlave()	스캔범위 내의 모든 슬레이브 장치들의 타입을 다시 체크하는 기능과 새로 연결된 슬레이브에 대한 체크
입력 / 출력	ReadInputWord()	지정한 슬레이브에 워드단위(16비트)로 데이터를 출력하는 기능
	WriteOutputWord()	지정한 슬레이브에서 워드단위(16비트)로 데이터를 입력받는 기능
	ReadInputByte()	지정한 슬레이브에 바이트단위(8비트)로 데이터를 출력하는 기능
	WriteOutputByte()	지정한 슬레이브에서 바이트단위(8비트)로 데이터를 입력받는 기능
	ReadInputBit()	지정한 슬레이브에 특정 비트에 ON 또는 OFF 값을 출력하는 기능
	WriteOutputBit()	지정한 슬레이브에 특정 비트의 값을 읽어 오는 기능

3. 결 론

본 연구에서는 PC를 주제어기로 사용하는 PCI HSIO(High-speed I/O) 시스템을 설계하고 제작하였다. 본 연구의 시스템은 기존의 방식과는 달리 분산방식의 입출력 구조를 가지면서 호스트의 부담을 줄여 고속으로 입출력이 가능하도록 설계되었다. 그러므로 매스터와 슬레이브 간의 통신 속도가 매우 빠르면서, 슬레이브의 확장성이 우수하고, 슬레이브에서 슬레이브로 연결할 수 있기 때문에 케이블 절감효과가 크다. 또한 각 슬레이브에서 이루어지는 통신들은 마스터가 전담하여 수행하므로 연산시간의 부담이 적은 특징을 갖는다.

[참 고 문 헌]

[1] 박한구외 1명, "제철공정에 PC-based 제어 시스템 적용 기술", Rolling 2001 : 제4회 압연심포지움, C65, 2001.

[2] 이종운외 1명, "USB 인터페이스를 갖는 산업용 IO제어기의 개발", 2001년도 대한전기학회 하계학술대회 논문집 D권, pp2362-2364, 2001.

[3] 조규상외 1명, "USB방식의 분산형 I/O 제어 시스템의 개발", 2004년도 대한전자공학회 하계종합학술대회 논문집 V권, 2004.

[4] High Speed Link Systems(HLS) User's Manual, StepTechnica Co., Ltd., 2002.

[5] "Programming the Microsoft Windows Driver Model", Walter Onely, 1999, MicroSoft Press.