論 文
55D-5-6

고속 DIO(Digital I/O) 시스템의 설계와 제작

Design and Implementation of a Fast DIO(Digital I/O) System

趙 奎 翔† · 李 鍾 云*
(Gyu-Sang Cho · Jong-Woon Lee)

Abstract - High speed PC-based DIO(Digital I/O) system that consists of a master device and slave I/O devices is developed. The PCI interfaced master device controls all of serial communications, reducing the load on the CPU to a minimum. The slave device is connected from the master device and another slave device is connected to the slave device, it can repeated to maximum 64 slave devices. The slave device has 3 types I/O mode, such as 16 bits input-only, 16 bits output-only, and 8bits input-output. The master device has 2 rings which can take 64 slaves each. Therefore, total I/O points covered by the master is 2048 points. The slave features 3 types of input/output function interchangeablility by DIP switch settings. Library, application, and device driver software for the DIO system that have a secure and a convenient functionality are developed.

Key Words : Digital I/O, Master-Slave I/O, PC-Based I/O System, G9001/G9002

1. 서 론

산업용 기기나 정밀도 높은 반도체 장비 분야 등과 화학이나 제철 공장과 같은 대규모 사업장에서 스위치 입력, AC 입력, 릴레이 출력, TR출력 등의 많은 수의 디지털 I/O를 필요로 한다. 이런 종류의 I/O의 처리는 주로 PLC (Programmable Logic Controller)[1]를 이용하는 경우가 많다. PLC를 사용하는 경우는 기기를 별도로 구입하는 비용과 PC와는 다른 운영시스템이 필요하고 설치 공간이 많이 필요하고, PC와 통신하는데 어려움이 많이 있으며, 케이블의 길이가 길어지는 단점이 있다. PC 기반의 I/O카드의 경우는 PLC의 여러 단점을 극복하면서 많이 활용되고 있지만 PC의 한정된 슬롯에 많은 수의 I/O 카드를 장착해야 하는 등의 문제점들이 있다[2].

최근 PC를 주 제어기로 사용하는 PC 기반의 I/O 시스템이 많이 사용되고 있다[3,4]. 이것은 실시간 제어성, 개발의 편리성, 신뢰성, 확장성 등의 측면에서 많은 장점이 있어 비용의 절감 효과가 있다. 이런 방식의 시스템에서는 마스터 장치에서 슬레이브로 연결되는 결선방식과 데이터 전송 속도, PC와 마스터 장치와의 인터페이스 방식이 주요한 성능의 척도가 된다.

마스터 장치에서 모든 슬레이브에 1:1로 연결하는 배선 방식이 일반적인 방법이지만 이런 방법은 PC의 슬롯에 많은 수의 I/O 카드를 장착해야 한다. USB를 사용하여 이런 단점을 보완한 연구[5]에서는 한 개의 USB 마스터 포트 당 4개의 슬레이브를 연결하여 최대 64 포인트를 사용할 수 있도록 설계되었다. 그러나 많은 포인트 수를 사용하는 경우에 추가적인 USB 마스터 장치가 필요한 것이 이 방식의 단점이다. 이런 단점을 개선한 연구[6]에서는 마스터 장치와 슬레이브 장치 간의 통신 속도가 매우 빠르며 배선을 절감할 수 있는 구조를 채택하여 마스터와 슬레이브를 분산 배치할 수 있는 구조의 USB HSIO(High Speed I/O)시스템을 개발하였다. 이 연구에 이어서 USB를 PCI방식으로 변경하여 좀 더 빠르고 안정되고 정해진 접근시간이 보장되는 PCI HSIO 시스템을 제작하였다[7].

본 연구에서는 PCI HSIO[7] 보다 속도, 사용가능 포인트 수, 설계의 간편성 등의 기능이 우수한 DIO(Digital I/O) 시스템을 설계 및 제작하기로 한다. 서론에 이어 본문에서 이것에 관한 구체적인 설계와 제작 방법에 대한 내용을 다룬다. 2장에서 전체적인 DIO 시스템의 개요를 설명하고 기존의 것과 구조와 성능에 대한 비교를 설명하고 3장에서는 간결한 결선 구조를 가지며 빠른 속도로 동작하는 마스터 장치와 슬레이브 장치에 대한 하드웨어 설계와 제작 방법에 대한 내용을 다룬다. 4장에서는 이 시스템에 사용되는 소프트웨어의 설계와 제작된 내용에 대하여 다룬다. 시스템의 구동에 필요한 디바이스 드라이버와 응용 프로그램의 제작에 필요한 라이브러리의 기능을 설명과 이것을 기반으로 제작된 응용 프로그램의 사례 등을 제시하고 5장에서 결론을 맺는다.

2. DIO 시스템의 개요

2.1 전체 시스템의 구성

DIO 시스템은 내부의 많은 램을 사용하여 슬레이브의 I/O를 마치 내부의 메모리에 접근하는 것과 같은 방식으로

† 교신저자, 正會員 : 동양대학교 컴퓨터학부 副敎授 · 工博
E-mail : cho@dyu.ac.kr
* 正 會 員 : 동양대학교 철도운전제어학과 副敎授 · 工博
接受日字 : 2006年 2月 10日
最終完了 : 2006年 3月 26日

제어한다. 이것은 CPU의 부담을 최소화 할 수 있도록 슬레이브 제어에 대한 부하를 줄여서 많은 수의 슬레이브 들과 빠르게 시리얼 통신하도록 설계된 것이다. 이로 인해 전체적으로 시스템이 간단한 형태로 구성되며, 결선방식의 특징으로 인해 케이블링 절감 효과를 얻을 수 있다. 또한 많은 수의 I/O 터미널을 쉽게 제어할 수 있어서 많은 데이터를 취급하기에 용이한 구조를 갖는다[8].

그림 1은 DIO 시스템을 하드웨어 부분과 소프트웨어 부분으로 구분하여 전체 시스템의 개요를 나타낸 것이다. 하드웨어는 마스터 장치와 슬레이브 장치로 구성된다. 마스터 장치는 PCI 인터페이스를 통해서 PC의 슬롯에 장착된다. 슬레이브는 마스터에 연결 케이블을 통해서 연결하고 슬레이브에서 다른 슬레이브로 연결하는 방식을 사용한다. PCI 통신을 위한 칩은 PLX 9030[9]이 사용되고, 마스터 장치에는 G9001, 슬레이브 장치에는 G9002가 사용된다.

마스터 장치에 장착된 G9001 한개는 최대 64개의 슬레이브 장치를 연결할 수 있다. 이 칩은 마스터 장치에 두 개 장착되는데 한쪽은 링(ring) 0이라고 부르고 다른 한쪽은 링 1이라고 부른다. 그러므로 최대 2x64=128개의 슬레이브 장치를 연결해서 사용할 수 있다. 마스터와 슬레이브 간의 통신은 Half-duplex 방식의 RS-485통신 방식을 사용하며, 통신 속도는 최대 20Mbps이다. 64개의 슬레이브가 연결된 경우에 모든 슬레이브에 대한 전송 사이클 시간(transfer cycle time)은 1ms이내가 된다.

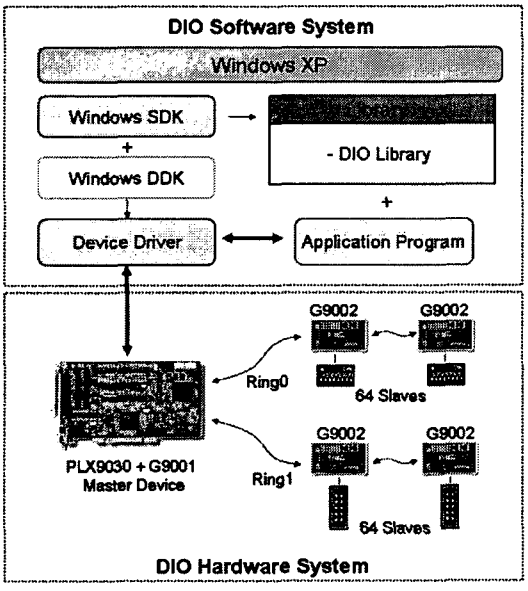

그림 1 DIO 시스템의 전체 구성도

Fig. 1 Configuration map of the overall system

슬레이브 장치마다 4포트(1포트=8포인트)의 입-출력이 지원되는데 목적에 따라서 입력 또는 출력으로 선택적으로 사용할 수 있다. 본 연구에서 개발한 슬레이브의 경우는 입력 전용으로 2포트, 출력 전용으로 2포트, 상위-입력 1포트 하위-출력 1포트, 상위-출력 1포트 하위-입력 1포트를 사용하는 방식으로 DIP스위치의 선택에 의하여 4가지 방식 중의 한 가지를 선택적으로 사용할 수 있도록 설계된다.

소프트웨어는 Windows XP에서 동작하도록 작성된다. 하드웨어를 구동하기 위한 디바이스 드라이버 프로그램은 Windows XP용 DDK를 사용하고 라이브러리와 응용프로그램 제작을 위해 SDK는 Visual C++ 6.0을 사용한다.

2.2 결선방식

본 시스템의 결선 방식은 그림 2에서처럼 마스터에서 슬레이브로 연결하고 슬레이브에서 다른 슬레이브로 연결하는 방식이다. 병렬 I/O 방식은 응답이 빠른 장점이 있으나 배선이 많아서 마스터와 슬레이브 간의 연결이 많아질 경우 많은 배선을 해야 하는 결정적인 단점이 있다. 직렬 I/O방식은 거리가 멀어도 직렬 통신을 이용하므로 선의 수가 적어서 배선에 유리한 점이 있지만 호스트에서 모든 통신 연산의 부담을 가지므로 속도 면에서 불리하다. 본 연구의 방식은 병렬의 장점과 직렬의 장점을 채택한 방식이다. 직렬-병렬 변환기능과 통신을 관장하는 전용 칩을 사용하여 마스터 장치의 입장에서는 슬레이브들이 내부의 메모리로 간주되어 통신을 위한 연산의 부담이 적다. 이 결선 방식은 매 슬레이브 장치마다 마스터 장치로 직접 연결하는 기존의 방식에 비하여 전체 배선의 길이가 짧아지고 슬레이브 장치를 원하는 위치에 배치할 수 있는 분산구조를 가지므로 비용의 절감과 공간 배치에 있어서 장점이 있다.

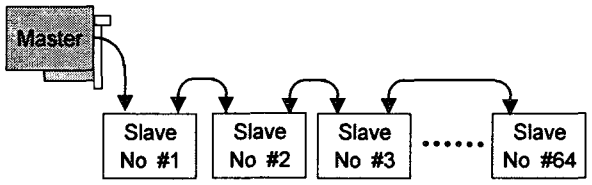

그림 2 마스터와 슬레이브의 결선 방식

Fig. 2 Master device to slave device wiring

2.3 기존의 시스템과의 방식 및 성능 비교

최신형으로 설계된 본 연구의 DIO 시스템 이전에 발표된 방식에 대하여 서론에서 간단히 언급한 것들과 주요 사항에 대한 성능 및 방식에 대한 비교를 하기로 한다. 본 연구의 방식을 "DIO", 참고문헌[5]의 경우를 "산업용IO", 참고문헌[6,7]의 경우를 "HSIO"라고 표현하기로 한다.

표 1에서의 첫 번째 항목은 "PC와의 인터페이스 방식"에 대한 비교이다. HSIO의 경우는 인터페이스 방식을 두 가지로 제작되었지만 마스터-슬레이브 간의 연결 방식은 동일하다. 두 번째 항목은 "데이터 통신 속도"를 나타내고 있다. 산업용 I/O의 경우는 USB1.1의 통신속도를 의미하는 것이고, HSIO와 DIO는 시스템 자체의 마스터-슬레이브 간의 통신 속도를 의미한다. 다섯 번째 항목의 "최대 가능 포인트 수"는 세 번째 항목과 네 번째 항목의 곱셈에 의하여 계산된 것이다. 여섯 번째 항목은 마스터와 슬레이브 간의 통신 방식을 나타내고 있다. 산업용I/O는 RS232, HSIO는 RS422, DIO는 RS485방식을 사용한다. 일곱 번째 항목의 통신 에러 체크의 방식은 3가지의 경우 동일한 방식을 사용한다. 여덟 번째 항목의 "슬레이브의 적용성" 항목은 제작된 슬레이브

의 종류와 사용의 확장성을 의미한다. 이것은 슬레이브의 제작의 간결성과 사용 방법이 얼마나 편리한 가에 따라 적용성이 결정된다.

3가지 시스템의 항목별 비교를 종합하여 보면 DIO 시스템이 기존의 두 방식에 비해서 통신의 속도, 슬레이브의 I/O 포인트 수, 슬레이브의 적용성 면에서 모두 우수한 것으로 평가된다.

표 1 기존의 시스템과 성능 및 방식 비교
Table 1 Comparison of DIO, and other systems

방식 / 항목	산업용IO	HSIO	DIO
PC와의 인터페이스 방식	USB	PCI/USB	PCI
데이터 통신 속도	12Mbps (USB1.1)	12Mbps	20Mbps
마스터당 최대 슬레이브 연결수	4	63	64
링의 수	1	1	2
최대 가능 포인트 수	4x16=64	63x16 =1,008	64x16x2 =2,048
마스터-슬레이브 연결 방식	RS232	RS422	RS485
통신에러체크 방식	CRC12	CRC12	CRC12
슬레이브의 적용성	표준화된 슬레이브 형식이 없음 필요에 따라 특수 제작	3종의 표준형 슬레이브 각기 제작 (입력전용, 출력전용, 입출력 겸용)	1종 슬레이브에서 3타입 사용가능

3. DIO 시스템의 하드웨어

DIO 시스템의 하드웨어는 마스터 장치와 슬레이브 장치로 구성된다. 마스터 장치는 PC와 통신을 위해 PCI 인터페이스가 장착되고 슬레이브를 제어할 수 있는 기능으로 구성된다. 슬레이브 장치는 입력과 출력을 위해 사용되는데 입력과 출력의 용도에 따라 입력전용, 출력전용, 입력-출력의 3가지 형태로 스위치를 절환하여 사용한다.

3.1 마스터 장치의 구성

마스터 장치는 PC의 슬롯에 장착되어 슬레이브 장치들을 제어하는데 사용된다. 이 마스터 장치에 두 개의 G9001[8]이 장착되는데 한개 당 최대 64개씩의 슬레이브 장치를 동시에 제어하는 것이 가능하다(64x2=128 슬레이브). 두 개의 칩에서 한쪽의 칩에 의한 마스터와 슬레이브 간의 관계를 ring0이라고 칭하고 다른 한쪽의 것을 ring1이라고 한다. PCI 통신을 위해 PLX9030(PLX technology)칩을 사용한다. 그림 3은 마스터 장치와 슬레이브 장치의 하드웨어 구성도이다.

3.2 마스터 장치의 메모리 맵

마스터 장치에는 슬레이브 장치를 제어하기 위한 각종의 정보들이 메모리에 기록되는데 슬레이브에 대한 정보를 마스터 장치에서 갖게 됨으로써 매우 빠르게 슬레이브를 제어할 수 있게 된다. 슬레이브 장치는 자체로 정보를 보유하지 않고 입출력의 단순 기능만을 담당한다. 슬레이브에 대한 각종의 정보가 마스터 장치의 메모리에 기록된다. 표 2는 마스터 장치의 메모리에 기록되는 슬레이브에 대한 정보 및 슬레이브 제어를 위한 정보들을 나타낸다.

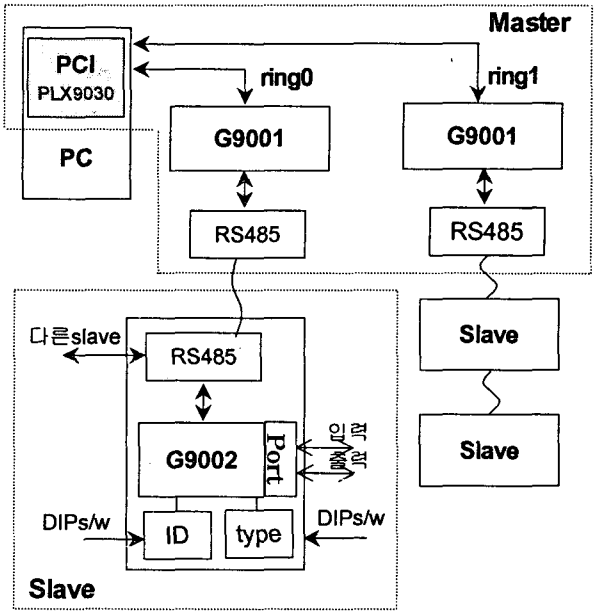

그림 3 마스터 장치와 슬레이브 장치의 구성도
Fig. 3 Block diagram of the master and slave device

표 2 마스터 장치의 메모리 맵
Table 2 The memory map of the master device

Address	쓰기	읽기
000-001	Command	status
002-003	Invalid	Interrupt status
004-005	I/O buffer	I/O buffer
006-007	Data sending FIFO	Data receiving FIFO
008-077	not specified	not specified
078-0B7	Device Info. 디바이스당 8bit. 0번에서 63까지 사용	Device Info. 디바이스당 8bit. 0번에서 63까지 사용
0B8-0BF	I/O 통신에러 플래그. 디바이스당 1bit사용	I/O 통신에러 플래그. 디바이스당 1bit사용
0C0-0FF	Input port change Interrupt settings 디바이스당 4bit사용	Input port change Interrupt settings 디바이스당 4bit사용
100-1FF	I/O port data 디바이스당 4바이트. 디바이스의 타입에 따라 입력/출력으로 사용	I/O port data 디바이스당 4바이트. 디바이스의 타입에 따라 입력/출력으로 사용

3.3 명령 코드

마스터 장치와 슬레이브 장치를 제어하기 위한 명령들은

마스터 장치의 000-001h번지까지의 16비트 크기의 명령어 데이터를 기록하면 해당하는 명령이 실행된다. 그 후에 그 번지에는 상태 비트 값들이 기록된다. 명령들은 리셋의 기능, 슬레이브에 대한 시스템 통신, I/O 통신의 시작과 종료, Data 통신의 시작과 종료 기능 등으로 구성된다. 표 3은 명령 코드값과 그에 해당하는 기능을 나타낸 것이다.

표 3. 명령어 코드표
Table 3 Command Codes

Command	기능
0000h	NOP
0100h	소프트웨어 리셋
1000h	모든 슬레이브에 시스템 통신 수행
1100h	특정 슬레이브를 제외한 모든 슬레이브에 시스템 통신 수행
1200h-123Fh	해당하는 비트에 의해 지정된 장치와 통신수행 0001 0010 00## ####(#ID비트지정)
3000h	I/O통신 시작
3100h	I/O통신 종료

3.4 슬레이브 장치의 구성

DIO 슬레이브 장치에는 G9002이 칩이 사용된다. 이것의 타입의 선택은 4가지 조합이 가능하다(표 4). 모드의 선택을 위해 딥 스위치를 장착하여 편리하게 선택할 수 있도록 설계된다. 입출력을 위한 데이터는 4포트가 할당되어 있다. 출력 전용으로 2포트, 입력 전용으로 2포트, 상위출력-하위입력을 각각 1포트, 상위입력 하위출력 각각 1포트씩 사용할 수 있도록 설계한다. 한 슬레이브는 4포트까지 사용할 수 있지만 입력에 2포트를 할당하고 출력에 2포트를 고정적으로 할당하는 방식을 사용하여 동시에 2포트를 사용할 수 있도록 설계한다. 기존의 연구에서 사용하는 슬레이브는 용도에 따라서 모두 다른 형태로 슬레이브를 제작해야 하지만 본 연구의 슬레이브들은 한 종류를 제작하여 타입을 달리 사용할 수 있는 다양한 적용성에 장점이 있다.

그림 4는 슬레이브 타입과 고유번호를 설정하기 위해 사용하는 모든 슬레이브마다 장착된 딥 스위치이다. 상위의 6개의 스위치는 슬레이브의 고유번호를 지정하는데 사용한다. 번호는 0에서 63번까지 부여할 수 있다. 링0과 링1의 고유번호는 같은 방식으로 번호를 부여하여 사용하고 서로 독립적으로 운영이 되기 때문에 같은 번호가 사용되어도 상관없지만 한 링 안에서는 중복된 번호를 사용해서는 안 된다.

3.5 슬레이브 정보

마스터 장치의 메모리에 슬레이브에 대한 정보는 0번에서 63번까지의 모든 슬레이브를 순회 통신을 한 후에 기록된다. 슬레이브의 응답에 따라 마스터 장치의 연결확인, 디바이스 타입, I/O 포트에 대한 설정상태 등을 주기적으로 갱신한다. 슬레이브의 타입은 표 4에 나타낸 바와 같은 4가지 방식이 가능하다. 타입의 번호는 셋팅 값에 의하여 정해진다. 각 셋팅은 딥스위치에 의하여 설정되도록 만들어져 있다. 이것의 설정에 따라 사용하는 포트가 결정되고 입력 또는 출력의 기능을 하도록 설계되었다.

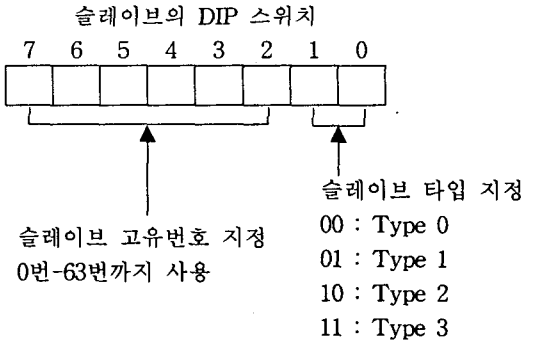

그림 4 슬레이브 ID와 타입 지정을 위한 스위치 설정
Fig. 4 ID and type settings for the slave device

마스터 장치에 슬레이브들이 연결되면 078h-0B7h까지의 메모리 번지에서 각 슬레이브 번호에 해당하는 메모리 어드레스에 그림 6과 같은 8비트 정보가 기록되어 표4에 나열된 슬레이브 셋팅에 해당하는 값으로 저장된다.

표 4 슬레이브의 타입과 포트번호
Table 4 Type and port number of the slave device

타입 번호	셋팅	Port0	Port1	Port2	Port3
type0	000	x	x	Output	
type1	001	Input	x	x	Output
type2	010	x	Input	Output	x
type3	011	Input		x	x
x	1xx	unspecified			

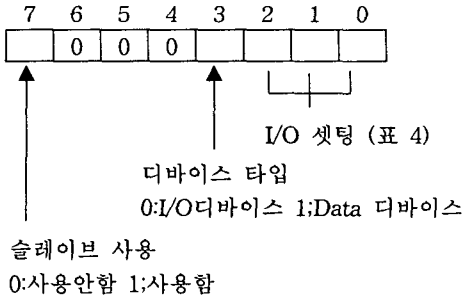

그림 5 마스터 메모리에 저장되는 슬레이브 정보
Fig. 5 Slave Information section format of the master memory

그림 5의 7번 비트는 슬레이브가 연결되었을 때에 1값이 되고, 연결되지 않은 경우에 0값이 된다. 이 비트를 사용하여 현재 연결된 슬레이브들에 대한 상태를 바로 체크할 수 있다. 3번 비트는 I/O디바이스와 Data 디바이스를 구분하는데 사용한다. 본 연구에서 제작된 슬레이브들은 모두 I/O 디바이스 타입이므로 이 값은 항상 0이 된다.

그림 6과 7은 각기 제작된 마스터 장치와 슬레이브 장치의 실물사진이다. 두개의 슬레이브에 대하여 동작 실험을

하고 있다. 입출력 상태를 LED에 점등하여 동작중임을 나타내고 있다.

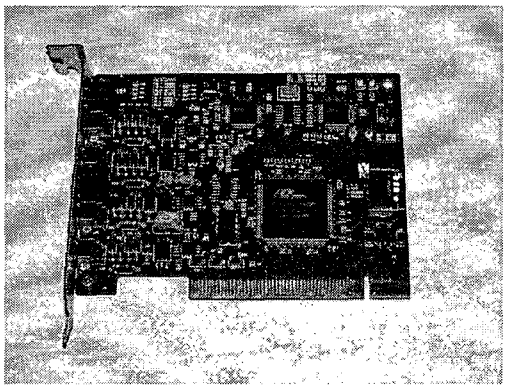

그림 6 제작된 마스터 장치
Fig. 6 The master device sample

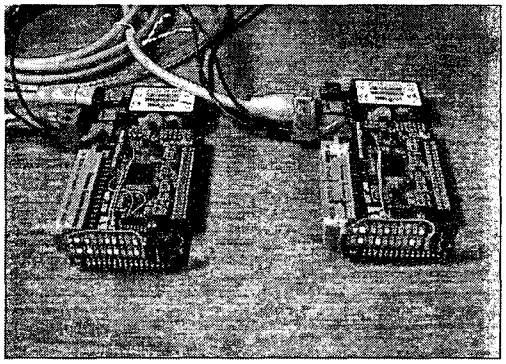

그림 7 제작된 슬레이브 장치
Fig. 7 The slave device samples

4. DIO 시스템의 소프트웨어

4.1 소프트웨어 구성과 개발도구

Windows XP용 디바이스 드라이버는 WDM(Windows Driver Model)[10] 방식으로 제작한다. 디바이스 드라이버의 개발도구는 Microsoft에서 제공하는 Windows XP용 DDK(Device Development Kit)를 사용한다. 작성된 드라이버는 DIO.sys이고 PnP 장치관리자에 의하여 c:\Windows\system32\dirvers에 설치된다. 설치 정보는 DIO.inf 파일에 작성된다.

라이브러리와 어플리케이션 제작을 위해 필요한 SDK (Software Development Kit)는 Visual C++ 6.0을 사용한다. 라이브러리를 제공하기 위한 파일은 헤더파일 DIO.h, 라이브러리 파일 DIO.lib, 동적링킹 라이브러리 DIO.dll로 제공한다. 어플리케이션은 라이브러리의 활용을 극대화하기 위한 샘플로 제작되는데 마스터의 기능과 슬레이브의 기능을 다이얼로그 방식의 프로그램 HSIOdlg.exe가 제작된다. 어플리케이션과 라이브러리 개발의 디버깅 도구는 Visual Studio의 디버거를 사용되고 디바이스 드라이버의 디버깅 도구는 WinDBG가 사용된다.

4.2 라이브러리 제작

DIO 시스템 제어를 위해 제작된 라이브러리는 크게 PCI 시스템의 설정과 마스터 장치의 초기화, 슬레이브 제어, 입력/출력, 링설정/해제 등의 4가지 기능으로 구분된다(표 5).

PCI 초기화와 마스터 장치의 초기화를 위한 기능은 5종의 함수가 제공된다. InitializePCI()함수는 DIO.sys로 작성된 PCI 드라이버를 개방하여 드라이버와의 통신을 준비한다. 지정된 드라이버가 설치되어 있지 않으면 에러 메시지를 출력하고 초기화 과정을 중지한다. 이 함수와 같이 사용되는 함수는 FinishPCI()함수인데 이 함수는 드라이버의 사용을 종료하기 위해서 호출된다. StartIOComm()과 StopIOComm()은 초기화가 이루어진 상태에서 마스터 장치와의 통신을 시작하거나 중단하는데 사용된다. 이 함수를 호출한 후부터 자유롭게 슬레이브들에 대한 입출력 명령이 수행될 수 있다. Reset()을 호출하면 모든 설정치가 초기화된다. 처음 프로그램을 시작할 경우나 입출력에 오류가 발생한 경우에 이것을 호출한다.

슬레이브 제어를 위한 함수 4개가 제공된다. 이중에서 GetDeviceType()은 인수에 슬레이브 번호를 지정하면 해당 슬레이브의 타입 값이 리턴되는 함수이다. 슬레이브의 타입에 따라 4포트 중에서 2포트를 출력 또는 입력으로 사용하는데 그 선택의 조합은 표 4의 Port0~Port4에 나타낸 바와 같다. GetAllDeviceType()는 0-63번까지의 모든 슬레이브에 대한 검사를 수행하여 인수로 지정한 정수형 포인터로 지정된 배열에 현재 연결되어 있는 슬레이브의 타입 정보를 저장한다. GetConnectedDevice()는 현재 연결되어 있는 슬레이브 장치의 번호와 타입을 알려주는 기능을 수행한다. 인수로 지정된 구조체의 포인터 위치에 슬레이브 번호와 타입을 저장하고 현재 연결되어 있는 슬레이브의 수를 리턴한다. ClearAllSlave()함수는 마스터 장치에 기록되어 있는 모든 슬레이브의 정보를 지우는 기능을 수행한다. 사용가능 모드는 2종류인데 mode=CLEAR_SLAVEMOEMORY는 입력포트와 출력포트의 내용을 클리어하는 기능을 수행하고, mode=CLEAR_ALLWITHRESET을 수행하면 Reset()함수를 호출한 후에 입력과 출력의 메모리를 클리어 한다.

입력과 출력용 함수는 8개가 제공된다. 이 함수들은 기능에 따라서 입력용 함수가 4종, 출력용 함수가 4종이다. 각각은 워드 단위 입출력, 바이트 단위 입출력, 워드 단위 데이터 중 비트 단위 입출력 , 바이트 단위 데이터 중 비트 단위 입출력 기능으로 세분된다.

WriteWord()는 워드 단위의 출력을 하기 위한 함수이다. 첫번째 인수에 슬레이브 번호, 두번째 인수에 16비트 데이터를 지정하면 해당 값들이 출력으로 나온다. WriteByte()는 바이트 단위의 출력을 하기 위한 함수이다. 첫번째 인수에 슬레이브 번호, 두번째 인수에 슬레이브 타입, 세번째 인수에 8비트 데이터를 지정하면 1바이트 출력이 수행된다. WriteWordBitwise()는 워드 단위 데이터 중에서 한 비트 부분만 On,Off시키는 기능을 수행한다. 첫번째 인수에는 슬레이브 번호, 두번째 인수에는 비트의 위치, 세번째 인수에는 0 또는 1을 지정한다. WriteByteBitwise()는 바이트 단위 데이터 중에서 특정비트를 On,Off시키는 기능을 수행하는데, 첫번째 인수에 슬레이브 번호, 두번째 인수에 슬레이브 타

입, 세번째 인수에 비트 위치, 네번째 인수에 0 또는 1을 지정한다. 입력용의 함수들 중에서 ReadWord()는 인수로 지정한 슬레이브에서 워드 단위의 값이 리턴된다. ReadByte()는 슬레이브 번호와 타입을 인수로 지정하면 바이트 단위의 값이 리턴된다. 워드 단위의 입력 중에서 인수로 지정한 특정 비트의 값을 ReadWordBitwise()로 구할 수 있고, ReadByteBitwise()로 바이트 단위의 입력 중에서 인수로 지정한 특정 비트의 값을 읽을 수 있다

표 5 라이브러리 함수명과 기능 설명
Table 5 Library functions and the descriptions

함수명	기능 설명
PCI 시스템의 설정과 마스터 장치의 초기화	
InitializePCI()	PCI카드의 초기화를 위한 함수
FinishPCI()	PCI 카드의 사용을 종료함
StartIOComm()	슬레이브와 통신을 시작하는 명령
StopIOComm()	슬레이브와 통신을 종료하는 명령
Reset()	마스터 장치를 리셋
슬레이브 제어	
GetDeviceType()	마스터에 연결되어 있는 슬레이브 장치의 타입을 정수값으로 리턴
GetAllDeviceType()	모든 슬레이브에 대한 타입정보를 인수로 지정한 구조체에 값을 저장
GetConnectedDevice()	현재 연결된 슬레이브에 대한 타입 정보만을 인수로 지정한 구조체에 값을 저장
ClearAllSlave()	마스터카드의 메모리에 남아 있는 모든 슬레이브의 입출력된 내용을 클리어
입력/출력	
WriteWord()	지정한 슬레이브에 워드단위(16비트)로 데이터를 출력
WriteByte()	지정한 슬레이브에서 바이트단위(8비트)로 데이터를 출력
WriteWordBitwise()	지정한 슬레이브에 현재 출력되어 있는 워드단위의 데이터 중에서 지정한 비트만 바꿈
WriteByteBitwise()	지정한 슬레이브에 현재 출력되어 있는 바이트단위의 데이터 중에서 지정한 비트만 바꿈
ReadWord()	지정한 슬레이브에 워드단위로 데이터를 읽어 옴
ReadByte()	지정한 슬레이브에서 바이트단위 데이터를 읽어 옴
ReadWordBitwise()	지정한 슬레이브에 입력되는 워드단위의 데이터 중에서 지정한 비트의 데이터를 읽어 옴
ReadByteBitwise()	지정한 슬레이브에 입력되는 바이트단위의 데이터 중에서 지정한 비트의 데이터를 읽어 옴
링	
SetRing()	링의 번호를 정수값(0또는1)로 설정
GetRing()	현재 설정된 링의 정수값을 구함

DIO시스템에는 두 개의 G9001칩이 장착되어 서로 독립된 제어를 수행한다. 각각 ring0과 ring1로 정의한다. SetRing()은 인수에 링의 번호를 지정하여 현재의 사용 링을 설정한다. GetRing()는 현재 설정되어 있는 링의 번호를 정수로 리턴한다. SetRing()에 의해 지정된 링의 번호는 새로운 설정을 하기 전까지 계속 유효하다.

4.3 응용 프로그램

응용 프로그램은 제공된 라이브러리를 사용하여 마스터 장치와 슬레이브 장치의 모든 기능을 구현한다. 프로그램은 HSIOdlg.exe로 제작된다. 전체 시스템의 상태를 나타내고 제어하기 위한 마스터 장치는 그림 8과 같이 아이콘의 리스트 형태로 현재 연결된 슬레이브들의 정보를 표시한다. 여기에는 슬레이브 번호 타입 정보가 표시되고 타입에 따라 다른 형태의 아이콘이 표시된다. 이것 중의 하나를 선택하여 더블 클릭하면 해당하는 슬레이브의 제어창이 나타난다. 그림 9는 슬레이브 타입 1의 창이 동작하고 있는 것이다. 슬레이브 타입, 슬레이브 고유번호, 입출력 비트 수 등 슬레이브 정보를 나타내고 있다. 입력과 출력의 기능을 수행하기 위하여 체크박스로 각 비트들의 입력과 출력 설정을 표시하고 "읽기", "쓰기"의 푸쉬버튼을 누르면 동작하도록 제작되었다.

이 응용 프로그램은 입력과 출력의 기능을 보여주기 위한 예제 형식의 프로그램으로 제작되었다. DIO의 라이브러리를 사용하면 사용자의 제어 대상 형태와 응용 방법에 따라 매우 다양한 형태로 입출력 제어 프로그램을 제작할 수 있다.

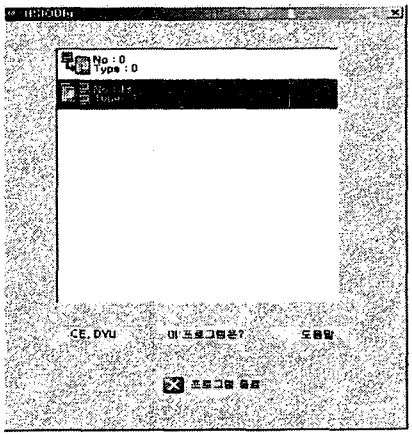

그림 8 마스터장치 창
Fig. 8 The master device window

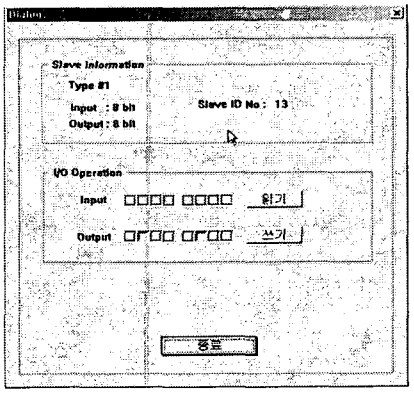

그림 9 슬레이브 장치-타입1
Fig. 9 The slave device-type 1

5. 결 론

본 연구에서는 마스터와 슬레이브 형식의 구조를 갖는 고속 DIO(Digital I/O) 시스템을 설계 및 제작하였다. 마스터 장치는 PCI 인터페이스 방식을 사용하는 PC 기반의 I/O 시스템을 구현하였다. 슬레이브 장치는 마스터에서 슬레이브로, 슬레이브에서 다른 슬레이브로 연결하는 결선 방식을 사용하여 분산배치가 가능한 형태이고 기존의 시스템이 갖고 있는 길고 복잡하게 연결되는 케이블링 문제를 해결하였다. 본 시스템은 마스터 장치의 메모리에 슬레이브의 입출력과 속성 정보 등을 가짐으로써 매우 빠른 입출력이 가능하게 되었고, 구조가 간편하게 되었다. 본문의 2.3절의 표 1에 명시한 바와 같이 이 DIO 시스템은 통신의 속도, 슬레이브의 I/O 포인트 수, 슬레이브의 적용성 면에서 기존의 방식에 비하여 모두 우수하다. 또한 슬레이브의 포트를 입력과 출력으로 선택적으로 사용할 수 있는 구조로 설계되었기 때문에 기존의 것에 비해서 설계 비용과 제작 비용을 절감하고 사용자의 편의성을 높였다.

소프트웨어는 Windows XP에서 동작하도록 작성되었다. WDM 방식의 디바이스 드라이버를 제작하였고, DIO 시스템의 PCI 초기화 설정과 마스터 장치의 초기화, 슬레이브 제어, 입력/출력, 링설정/해제 등의 기능을 갖는. 라이브러리 소프트웨어를 제작하였고 이것을 사용하는 활용 예제로써 다이어로그 방식의 어플리케이션 프로그램을 제작하였다.

PC를 기반으로 하는 I/O 제어기의 일반화와 PLC의 소프트웨어화가 계속 진행되고 있는 상황에서 디지털 I/O가 필요한 정밀장비나 제어기기, 공정제어 등에 본 연구에서 설계 및 제작한 시스템을 적용하면 낮은 설치비용으로 높은 성능을 구현할 수 있고, 라이브러리 형태로 제공되는 기능을 이용하여 사용자의 환경에 적합한 입출력 제어를 위한 고기능의 프로그램의 제작이 가능하다.

참 고 문 헌

[1] Kelvin T. Erickson, "Programmable Logic Controller", IEEE Potential, February/March, pp. 14-17, 1996.

[2] M. Juniper, "PC vs PLC-The Debate Rages On", Control Systems -Tonbridge, vol. 13, no. 6, pp. 33-34, 1996.

[3] C. Agostino, et. al, "Automation of Direct Reduction Plant using PLC/PC", IFAC sysposium on intelligent components and instruments for control applications, pp. 341-346, 2001.

[4] 박한구외 1인, "제철공정에 PC-based 제어 시스템 적용기술", Rolling 2001 : 제4회 압연심포지움, C65, pp. 15-21, 2001.

[5] 이종운외 1인, "USB 인터페이스를 갖는 산업용 IO 제어기의 개발", 2001년도 대한전기학회 하계학술대회 논문집 D권, pp2362-2364, 2001.

[6] 조규상외 1인, "USB방식의 분산형 I/O 제어 시스템의 개발", 2004년도 대한전자공학회 하계종합학술대회 논문집 V권, pp. 1477-1480, 2004.

[7] 조규상외 1인, "PCI 방식의 HSIO(High Speed I/O) 시스템의 개발", 2004년도 전기학회 하계학술대회 논문집 D권, pp.2628-2630, 2004.

[8] "Motionnet User's Manual", ppI-1~IV-4,NPM, 2004.

[9] "PCI 9030 Data Book Version 1.4", PLX Technology, pp 1-198, May, 2002.

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

[11] 조규상외 1인, "고속 DIO 시스템의 하드웨어 설계 및 제작", 2005년도 전기학회 하계학술대회 논문집 D권, pp.3031-3033, 2005.

[12] 조규상외 1인, "고속DIO 시스템을 위한 라이브러리 소프트웨어 및 응용프로그램 개발", 2005년도 전기학회 하계학술대회 논문집 D권, pp.3034-3036, 2005.

저 자 소 개

조 규 상 (趙 奎 翔)

1963년 3월 31일생. 1986년 한양대학교 전자공학과 졸업. 1997년 한양대학교 대학원 전자공학과 졸업(공박). 1996~현재 동양대학교 컴퓨터학부 부교수

Tel : 054-630-1119

Fax : 054-630-1119

E-mail : cho@dyu.ac.kr

이 종 운(李 鍾 云)

1961년 4월 20일생. 1982년 서울대학교 전자공학과 졸업, 1993년 한국과학기술원 전기전자공학과 졸업(공박). 1998 ~ 현재 동양대학교 철도운전제어학과 부교수

Tel : 054-630-1205

Fax : 054-630-1205

E-mail : jwlee@dyu.ac.kr