2005년도 대한전기학회 하계학술대회 논문집 2005. 7. 18 - 20

고속 DIO시스템을 위한 라이브러리 소프트웨어 및 응용프로그램 개발

조규상*, 이종운**
동양대학교 컴퓨터학부*, IT전자공학부**

Development of Library and Application Software for a Fast DIO System

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

Abstract

High speed PC-based digital I/O system, PCI-bus master and slave set is developed, which features are distributed structure, input/output function interchangeability by switch settings, and high speed(20Mbps). Library and application software for a DIO system that have a secure and a convenient functionality are developed.

1. 서 론

최근 PC를 주 제어기로 사용하는 PC-based 시스템이 많이 사용되고 있다. 이것은 실시간 제어성, 개발의 편리성, 신뢰성, 확장성 등의 측면에서 많은 장점이 있어 비용의 절감 효과가 있다[1].

이런 방식의 시스템에서는 마스터 장치에서 슬레이브로 연결되는 결선방식과 속도, PC와 마스터 장치와의 인터페이스 방식이 주요한 성능 비교를 위한 지표이다.

PC에 장착한 마스터 장치에서 매 슬레이브로 배선하는 방식이 일반적인 방법이다. 이 방법은 I/O카드를 장착할 때에 좁은 PC의 슬롯에 많은 I/O 카드를 장착해야 하는 어려움이 있다. 이런 단점들을 극복하기 위한 방법으로 USB가 사용된다[2]. 이 연구[2]에서는 USB 포트당 4개의 슬레이브를 연결하여 최대 64 포인트까지 사용가능하지만 포인트 수가 많아지면 더 많은 USB 장치를 통한 연결이 필요한 것이 단점이다. 이 방식의 단점을 개선한 연구[3]에서는 마스터 장치와 슬레이브 장치 간의 통신 속도가 매우 빠르며 배선을 절감할 수 있는 마스터-슬레이브의 분산 배치가 가능한 시스템을 개발하였다. USB는 간편하지만 포트가 갑작스럽게 빠지거나 매번 케이블의 연결시마다 디바이스의 설정을 다시 하는 불편함이 있다. 이 시스템의 인터페이스를 PCI방식으로 대체한 시스템을 발표하였는데 매우 빠르고 안정적으로 동작한다[4].

본 연구는 연구[4]와 같은 결선 방식을 사용하지만 더욱 빠른 속도로 입출력이 가능한 시스템[5]을 개발하였고 이것의 소프트웨어에 대한 개발 사례를 본 연구에서 제시한다. 효율성이 있고 안정적인 소프트웨어를 제작하는 것을 목표로 디바이스 드라이버와 라이브러리 소프트웨어와 이것을 활용한 어플리케이션 프로그램 제작을 수행한다.

2. 본 론

2.1 시스템의 개요

DIO(Digital I/O) 시스템은 마스터 장치와 슬레이브 장치로 구성된 하드웨어 요소들과 디바이스 드라이버 프로그램, 라이브러리, 어플리케이션 프로그램 등의 소프트웨어 요소로 구성된다(그림 1).

2.1.1 DIO 시스템의 하드웨어 구성

마스터 장치는 PC의 슬롯에 장착하여 PCI 버스 방식을 사용하여 하드웨어와 통신을 수행한다. 슬레이브는 마스터에 연결 케이블을 통해서 연결하고 슬레이브에서 다른 슬레이브로 연결하는 방식을 사용한다. PCI통신을 위해 PLX9030을 사용한다. 이것은 마스터 장치에 G9001[6]과 함께 사용된다. G9001은 마스터 장치의 핵심 역할을 하며 슬레이브에 대한 정보를 갖고 있고 슬레이브를 제어하기 위한 기능을 수행한다. 본 DIO 시스템에서는 두개의 G9001을 사용한다. 한 개의 G9001에는 64개의 슬레이브 연결이 가능하다. 이것을 ring이라고 부른다. 두개의 링이 형성되는데 이것을 ring0와 ring1이라고 부른다.

슬레이브에는 입력과 출력을 위한 주변장치를 연결하기 위한 소켓이 장착되어 있어 여기에 외부 기기 및 센서 등을 연결하여 사용한다. 슬레이브 마다 한 개의 G9002[6]칩이 장착되는데 이것에 의해 입출력 가능이 수행된다. 각 슬레이브는 4포트(1포트=8bit)가 지원되는데 목적에 따라서 입력 또는 출력으로 선택적으로 사용된다. 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 방식으로 사용가능하다. 그리고 자신의 고유한 ID번호를 DIP스위치로 설정할 수 있도록 제작된다.

다음의 내용은 DIO 시스템이 갖는 특장점을 나타낸 것이다.

- 마스터 칩 당 64개 슬레이브 연결가능
- 한 PCI 카드 당 2개의 마스터 칩 장착(2 ring)
  =64(슬레이브)x2(링)x16(비트)=2048포인트
- 마스터-슬레이브, 슬레이브-슬레이브의 결선방식 채택으로 배선길이와 속도 면에서 장점 가짐
- 20Mbps의 속도로 통신
- RS485 방식의 통신(Half-duplex)
- 20Mbps로 통신하는 경우
  8개 device 는 0.12msec이내 통신완료
  16개 device 는 0.24msec이내 통신완료
  32개 device 는 0.49msec이내 통신완료
  64개 device 는 0.97msec이내 통신완료
- 케이블길이
  32개 슬레이브 연결시 최대 100m 가능
  64개 슬레이브 연결시 최대 50m 가능
- CRC12 방식의 에러검출방식 사용
- 마스터 장치에서 모든 시리얼 통신에 대한 기능을 제어하기 때문에 CPU의 부하가 줄어듦
- 많은 RAM용량으로 원격I/O에 대한 접근을 단순한 메모리의 접근으로 가능케 함

기존의 병렬 I/O 연결 방식은 응답이 빠르지만 배선이 많아 지는 단점이 있다. 기존의 직렬 I/O방식은 거리가

멀어도 직렬 통신을 이용하므로 선의 수가 적지만 호스트에서 모든 통신 연산의 부담이 집중되어 속도가 느린 단점이 있다. 이에 비해 본 DIO 시스템은 마스터에서 슬레이브로 연결하고 슬레이브에서 다른 슬레이브로 연결하는 방식(그림 1)을 사용한다. 이 방식에서는 직렬-병렬 변환기능과 통신을 관장하는 전용 칩을 사용하여 마스터에서는 메모리에 대한 연산을 수행하면 된다. 그러므로 배선길이와 속도 면에서 장점을 갖는다.

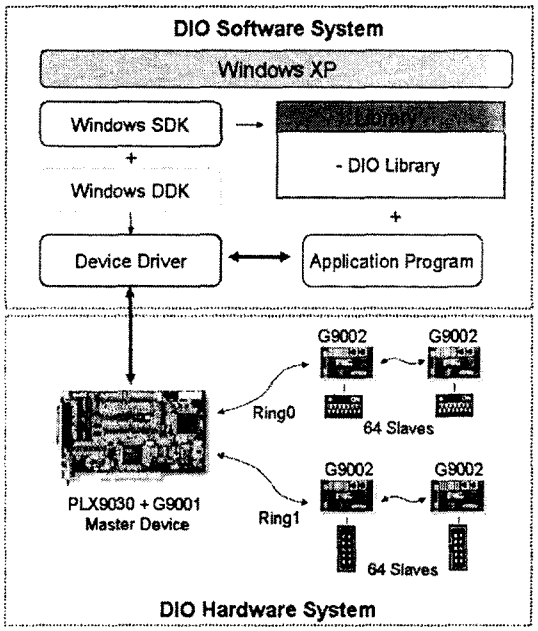

그림 1 DIO 시스템의 하드웨어와 소프트웨어 구성도

2.1.2 DIO 시스템의 소프트웨어 구성

Windows에서 하드웨어를 구동하기 위해서는 디바이스 드라이버가 필요하다. 이것은 WDM(Windows Driver Model)[7] 방식으로 제작한다. 디바이스 드라이버를 개발하기 위해서 Microsoft에서 제공하는 Windows XP용 DDK(Device Development Kit)를 사용해야 한다. 개발한 드라이버는 "PCI9030 WDM Driver"라고 장치관리자에 등록되며 드라이버 파일은 DIO.sys로 작성되고, 이것은 c:\Windows\system32\dirvers에 설치된다. 설치를 하기 위한 정보는 DIO.inf에 작성되어 있고, 여기에는 제공자의 이름과 제품명, 드라이버 명, 설치 경로, 설치파일명 등이 기록되어 있다.

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. 디버깅 도구는 어플리케이션과 라이브러리 개발에는 Visual Studio의 자체 디버거를 사용하고 디바이스 드라이버 프로그래밍 개발에는 WinDBG를 사용한다.

2.2 라이브러리

DIO 시스템 제어를 위해 제작된 라이브러리는 크게 PCI 시스템의 설정과 마스터 장치의 초기화, 슬레이브 제어, 입력/출력, 링설정/해제 등의 4가지 기능으로 구분된다. 표 1은 전체 라이브러리를 기능별로 구분하여 간략하게 정리해놓은 것이다. 간략하게 정리해놓은 것이다.

표 1 DIO시스템을 위한 라이브러리 함수명과 기능설명

함수명	기능 설명
PCI 시스템의 설정과 마스터 장치의 초기화	
InitializePCI()	PCI카드의 초기화를 위한 함수
FinishPCI()	PCI 카드의 사용을 종료함
StartIOComm()	슬레이브와의 통신을 시작하는 명령
StopIOComm()	슬레이브와의 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 워드 단위의 데이터 중에서 지정한 비트의 데이터를 읽어 옴
ReadByteBitwise()	지정한 슬레이브에 입력되는 바이트 단위의 데이터 중에서 지정한 비트의 데이터를 읽어 옴
링	
SetRing()	링의 번호를 정수값(0또는1)로 설정
GetRing()	현재 설정된 링의 정수값을 구함

2.2.1 시스템 초기화

PCI 초기화와 마스터 장치의 초기화를 위한 기능은 다음과 같이 5종의 함수가 제공되고 있다.

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()함수는 초기화가 이루어진 상태에서 최초로 마스터 장치와의 통신을 시작하거나 종료하는데 사용된다. 이 함수를 호출한 이후부터 정상적인 슬레이브에 대한 입출력 명령이 수행될 수 있다. Reset()함수를 호출하면 모든 명령과 설정치가 최기화 되므로 주로 처음 프로그램을 시작할 경우나 입출력에 오류가 발생한 경우에 수정용으로 호출하는 경우에 사용된다.

2.2.2 슬레이브 제어

슬레이브 제어를 위한 함수들은 GetDeviceType(), GetAllDeviceType(), GetConnectedDevice(), ClearAllSlave()등의 4종이 제공되고 있다.

GetDeviceType()함수는 인수에 슬레이브 번호를 지정하면 해당 슬레이브의 타입값이 리턴되는 함수이다. 슬레이브의 타입은 표2와 같이 정의된다[5]. DIO시스템의 슬레이브는 2바이트를 출력과 입력으로 사용하는데 그 중

에서 상위바이트와 하위바이트를 서로 다른 용도로 사용할 수 있다[표2].

표 2 타입번호와 입출력의 기능

타입번호	코드값	입출력 기능
Type 0	0x80	상위-출력, 하위-출력
Type 1	0x81	상위-출력, 하위-입력
Type 2	0x82	상위-입력, 하위-출력
Type 3	0x83	상위-입력, 하위-입력

GetAllDeviceType()함수는 0-63번까지의 모든 슬레이브 번호에 대한 검사를 수행하여 인수로 지정한 정수형 포인터로 지정된 배열에 데이터를 저장한다.

GetConnectedDevice()함수는 현재 연결되어 있는 슬레이브 장치의 번호와 타입을 알려주는 기능을 수행한다. 인수로 지정된 구조체의 포인터 위치에 슬레이브 번호와 타입을 저장하고 현재 연결중인 슬레이브의 갯수를 리턴하는 한다.

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.

2.2.3 입출력

입출력용으로 제작된 함수들은 8종의 함수가 제공된다. 이 8종의 함수들은 기능에 따라서 입력용 함수가 4종, 출력용 함수가 4종이다. 각각은 워드단위 입출력, 바이트 단위 입출력, 워드크기의 비트 단위 입출력 , 바이트크기의 비트단위 입출력 기능으로 세분된다.

WriteWord()는 첫번째 인수에 슬레이브 번호, 두번째 인수에 16비트 데이터를 지정하면 출력으로 전달된다. WriteByte()는 첫번째 인수에 슬레이브 번호, 두번째 인수에 슬레이브 타입, 세번째 인수에 8비트 데이터를 지정하면 1바이트 출력이 수행된다. WriteWordBitwise()는 워드 단위의 값 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 단위의 데이터 중에서 특정비트를 On,Off시키는 기능을 수행하는데, 첫번째 인수에 슬레이브 번호, 두번째 인수에 슬레이브 타입, 세번째 인수에 비트 위치, 네번째 인수에 0 또는 1을 지정한다.

입력용의 함수들은 워드, 바이트, 비트 단위의 기능으로 구분된다. ReadWord()의 경우는 인수로 지정한 슬레이브에서 워드단위의 값을 리턴받고, ReadByte()는 슬레이브 번호와 타입을 인수로 지정하면 바이트 값을 리턴받는다. 워드단위의 입력중에서 특정한 비트의 값을 ReadWordBitwise()로 구할 수 있고, ReadByteBitwise()로 바이트 단위의 입력중에서 인수로 지정한 위치의 비트값을 읽을 수 있다.

2.3.4 링 설정/해제

DIO시스템에는 두 개의 G9001칩이 장착되어 서로 다른 슬레이브 들이 엮인 링을 제어할 수 있다. ring0과 ring1로 정의된 두 링을 선택하는 함수는 인수에 링의 번호를 지정하여 SetRing()을 호출한다. GetRing()함수는 현재 설정된 링의 번호가 정수로 리턴된다. SetRing()으로 지정한 링의 번호는 새로운 설정을 하기 전까지 계속 유효하다.

2.3 응용 프로그램

응용 프로그램은 라이브러리의 활용도를 극대화하기 위한 방안으로 제작된 것이다. 프로그램의 이름은 HSIOdlg.exe이다. 그림 2는 시스템에 현재 연결되어 있는

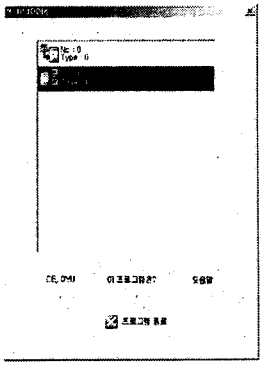

그림 2 마스터장치 창

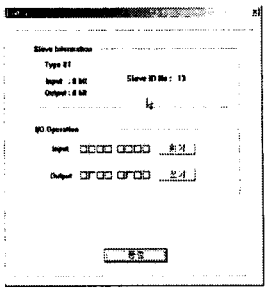

그림 3 슬레이트 타입1

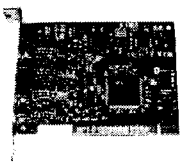

그림 4 마스터 장치

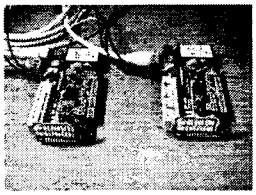

그림 5 슬레이브 장치

된 슬레이브 장치들이 ID와 타입의 값이 마스터 장치의 정보를 나타내고 있다. 선택된 리스트를 누르면 해당하는 슬레이브의 다이얼로그가 나타난다(그림3). 각 슬레이브의 다이얼로그에는 입력, 출력의 형태에 따라 버튼을 눌러 설정하거나 읽은 값을 표시하도록 작성되었다.

3. 결 론

본 연구에서는 PCI 인터페이스 방식을 사용하는 마스터 장치와 그와 분산형 구조로 배치가 가능한 결선방식을 사용하는 고속으로 입출력을 하는 DIO시스템을 설계하고 제작하고 이것에 대한 소프트웨어를 제작하였다. 소프트웨어는 디바이스 드라이버와 라이브러리 소프트웨어를 제작하였고 이것을 사용하는 활용 예제로써 다이어로그방식의 어플리케이션 프로그램을 제작하였다. 많은 실행 테스트를 통하여 소프트웨어들이 안정적으로 작동이 되는 것을 검증하였다.

[참 고 문 헌]

[1] 박한구 등, "제철공정에 PC-based 제어 시스템 적용기술", Rolling 2001 : 제4회 압연심포지움, C65, 2001.

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

[3] 조규상 등, "USB방식의 분산형 I/O 제어 시스템의 개발", 2004년도 대한전자공학회 하계종합학술대회 논문집 V권, pp. 1477-1480, 2004.

[4] 조규상 등, "PCI 방식의 HSIO(High Speed I/O)시스템의 개발", 2004년도 전기학회 하계학술대회 논문집 D권, pp.2628-2630, 2004.

[5] 조규상 등, "고속 DIO시스템의 하드웨어 설계와 제작", 2005년도 전기학회 하계학술대회 논문집, 2005.(예정)

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

[7] "Programming the Microsoft Windows Driver Model", Walter Oney, 1999, MicroSoft Press.