COM을 이용한 입출력 디바이스의 제어

황태문*, 이광규, 정갑식**, 최민희**, 이종혁****

*경성대학교 멀티미디어정보예술대학원 정보공학과

**경성대학교 컴퓨터공학과

Control of Input/Output Device Using COM

Tae-moon Hoang, Kwang-kyu Lee, Gab-sik Jung, Min-hee Choi, Jong-hyeok Lee

Kyungsung University

요 약

컴퓨터의 발달로 산업현장은 많은 부분이 자동화되고 있다. 자동화를 위해서 고려되어야 할 사항 중 하나가 입출력 디바이스 제어이다. 산업현장을 자동화하기 위해서 관련 프로그램 등이 필요하지만 이들 프로그램과 입출력 디바이스는 매우 종속적이다. 이에 본 연구에서는 COM을 이용하여 입출력 디바이스를 제어하고자 한다. 입출력 작업에 사용되는 범용 디바이스와 이를 컨트롤 할 수 있는 디바이스 드라이버를 개발하고, COM을 이용한 공장 자동화 프로그램을 개발하여 일반 사용자가 쉽게 그 기능을 사용할 뿐만 아니라 이들 프로그램을 재사용 할 수 있게 하였다.

주요어 : COM, PCI BUS, PCI Interface, 디바이스 드라이버, 컴포넌트

Ⅰ. 서론

현재 윈도우 응용 프로그램들이 채택하는 객체지향 설계(Object Oriented Design)기법은 데이터 부분까지를 캡슐화(Encapsulation) 시켜 객체(Object)간의 상호 관계를 최대한 줄임으로써, 각 객체들을 소프트웨어 부품으로 사용하려고 하였다. 그러나 이러한 캡슐화 기능은 이진(Binary) 객체들을 연결할 표준이 없고, 언어의 호환성이 까다로우며, 객체의 내용을 수정하는데 어려운 문제점으로 인하여 소프트웨어의 부품화를 충분히 달성하지 못하고 있다. 이러한 문제점을 해결하기 위하여 COM(Component Object Model)기법이 제안되었다.[1][2][3] COM을 사용하므로 사용자 입장의 경우 개발자가 만든 제어 프로그램을 바로 사용할 수도 있고 디바이스와 서버 프로그램만을 구입하여 직접 클라이언트 프로그램을 원하는 형태로 개발할 수 있으므로 최근 각광을 받고 있다.

한편, 기존의 산업현장에서 사용하는 ISA BUS는 좁은 대역폭과 느린 전송 속도로 인해 원활한 데이터의 전송을 할 수가 없다. 그래서 많은 양의 데이터를 빠른 시간에 전송하기 위해 PCI BUS와 연결할 수 있는 입출력 디바이스가 필요하다.[7]

본 연구에서는 PCI 입출력 디바이스와 이를 제어하는 드라이버를 개발하고, COM을 사용하여 프로그램을 개발함으로써 이를 제어할 수 있도록 하고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 디바이스에 대해 기술하고 3장에서는 디바이스 드라이브 개발을 기술한다. 4장에서 COM에 관한 내용을 기술하고 5장에서 COM을 이용한 입출력 디바이스 제어를 기술하고 그 실제 구현 예를 보인다. 마지막 6장에서 결론을 내리고 향후 과제를 제시하고자 한다.

Ⅱ. 디바이스

2.1 SYSTEM BUS

SYSTEM BUS는 CPU, 메모리와 더불어 PC를 구성하는 3대 요소의 하나로 PC내부에서 발생된 데이터가 이동하는 통로를 의미한다. 주변장치와 CPU, 메인 메모리와 CPU 등이 모두 버스를 통하여 데이터를 전송한다. 따라서 Bandwidth가 데이터의 전송률에 큰 영향을 미친다. 일반적인 시스템 버스를 그림 1 에 나타내었다.

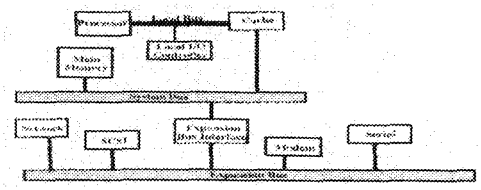

그림 95. 일반적인 SYSTEM BUS

PCI BUS는 8, 16, 32bit의 Data Length,

33MHz의 전송 속도와 다중 버스를 지원한다. PCI 버스가 장착된 마더보드는 PCI 버스가 하나가 아닌, 데이지체인처럼 계층적으로 여러 개 구성될 수 있으며, 심지어 PCI 버스가 아닌 다른 버스 시스템도 서브 버스로 구성할 수 있다.[5][8]

USB(Universal Serial Bus)는 느린 직렬, 병렬 인터페이스를 대체하기 위하여 개발된 인터페이스이지만 아직은 소프트웨어적인 지원이 미약하다.

2.2 PCI TARGET INTERFACE

개발한 하드웨어 보드를 PCI 버스에 연결하기 위하여 일반적으로 PCI TARGET INTERFACE CHIP을 이용한다. 인터페이스 칩을 사용하게 되면 INTERFACE CHIP에서 제공하는 로컬 버스를 이용하면 된다. 예를 들어 우리가 만든 하드웨어 보드가 그 특성상 8MHz까지의 속도로만 동작이 가능한 동기 회로라면 PCI 버스의 33MHz에 직접 연결해 사용할 수는 없다. 하지만 사용이 간편한 INTERFACE CHIP에서는 자신이 직접 PCI 버스와의 통신을 33MHz로 하더라도 자신이 하드웨어 보드에 제공하는 로컬 버스는 8 MHz 으로 설정해 하드웨어 보드를 동작시킬 수 있게 하는 기능을 제공해 준다. 대표적인 PCI TARGET INTERFACE CHIP으로 PLX 9050-1이 있으며 이의 블록도를 그림 2 에 나타내었다.

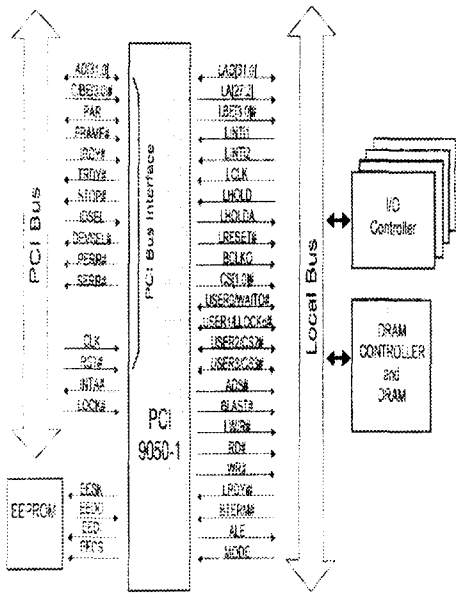

그림 2. 9050 Block Diagram

PLX 9050-1은 PCI 확장 보드를 마더 보드에 쉽게 연결할 수 있도록 해주며 PLX 9050-1은 다양한 local bus를 PCI BUS에 연결할 수 있도록 설계되어 있고 비교적 느린 local bus가 132MB/sec burst 전송이 가능하도록 설계할 수 있다. PLX 9050-1은 multiplexed 혹은 nonmultiplexed 8, 16, or 32-bit local bus에 연결하도록 프로그래밍 할 수 있다. 8/16bit mode에서는 ISA방식을 PCI 방식으로 쉽게 전환 할 수 있으며 PLX 9050-1은 느린 local bus를 32-bit, 33MHz의 속도를 가진 PCI BUS에 연결할 수 있도록 양방향 FIFO Buffer를 가지고 있다. 최대 5개의 local address space와 4개의 chip select register가 지원된다.

III. 디바이스 드라이브

3.1 Monolithic Driver

장치 드라이버는 자신의 컴퓨터에 부착된 특정 주변 장치들을 제어하기 위한 프로그램이다. 장치 드라이버는 본래 운영체제의 많은 일반적인 입출력 명령어들을 각 장치들이 이해할 수 있는 메시지의 형태로 변환하는 역할을 담당한다. 디바이스 드라이버의 종류는 Monolithic Driver, Layered Driver, Miniport Driver가 있다. Monolithic Driver는 하드웨어를 개발하여 사용하는 가장 기본적인 디바이스 드라이버이며 하나 이상의 사용자 응용프로그램에 의해 액세스되고 하드웨어를 직접적으로 제어한다. 드라이버는 IO 제어명령(IOC)을 통하여 응용프로그램과 통신하고 다른 DDK(Device Development Kit) 함수를 호출하여 하드웨어를 제어한다. 일반적인 Monolithic Driver의 블럭도는 그림 3과 같다.[4]

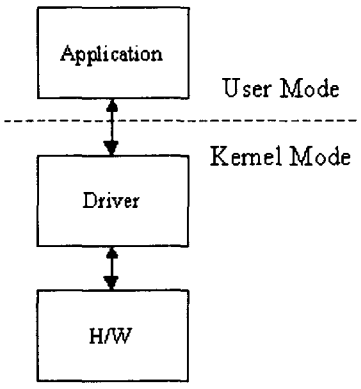

그림 97 Monolithic Driver

3.2 디바이스 드라이버 개발

WinDriver는 Monolithic Driver를 개발할 수 있는 툴이다. WinDriver는 커널 모드 디바이스 드라이버를 개발자가 코딩하지 않고 Win32 응용프로그램 안에서 직접 하드웨어를 액세스할 수

있게 한다. COM 프로그램 개발 시간 단축과 소스 내에 디바이스 제어 함수를 포함시키는 것을 원활하게 하기 위해 WinDriver를 이용하여 드라이버를 개발하였다.

WinDriver 내에 포함되어 있는 Driver Wizard를 사용하여 개발한 디바이스를 진단하고 기본 소스코드를 생성한다. 이 소스코드는 설정된 리소스를 액세스하고 인터럽트를 처리할 수 있는 기능을 가진 진단프로그램으로서 디바이스를 제어하는 데 필요한 함수들을 포함하고 있으며, 이들 함수를 이용하여 작성된 COM 프로그램은 디바이스를 제어 할 수 있다.

IV. Component Object Model

COM은 라이브러리, 응용/시스템소프트웨어, 여러 종류의 소프트웨어간의 상호작용을 가능하게 하는 공통된 컴포넌트 소프트웨어를 제작할 때 준수해야 할 여러 가지 규칙들의 집합이다. COM을 한마디로 표현하자면 "응용 프로그램과 컴포넌트간에 지켜져야 할 인터페이스 규격에 대한 모델(Model)"로 정의할 수 있으며 윈도우 환경에서는 이 모델을 통해 오브젝트간 통합이 가능하게 된다.[1]

COM의 특징은 객체 지향적이며 서로 동적으로 연결되어 있고 이런 각 컴포넌트는 전체 애플리케이션을 재 컴파일 할 필요 없이 변경 가능하다. 또한 COM은 이진표준이므로 여러 가지 언어로 작성될 수 있다.

COM 컴포넌트의 구조는, 서버(Server)에 하나 이상의 COM 객체를 포함하고 있으며 COM 라이브러리로 제공되는 서비스와 API로 갖추어져 있다.[8] COM은 다른 모듈에 있는 컴포넌트들이 서로 이야기할 수 있도록 통신 메카니즘을 제공하는데 이런 컴포넌트 서버의 종류로는 인 프로세스(In Processor) 서버(DLL 서버), 로컬 서버(Exe 서버) 그리고 리모트 서버(네트웍 요소가 포함된 Exe 서버) 등이 있다. 인 프로세스 서버는 서버와 클라이언트가 동일한 프로세스 내부에 존재하면서 상호 작동하는 것을 말한다. 로컬서버와 리모트 서버는 서버와 클라이언트가 각각의 프로세스에 따로 존재하는 서버를 말하는데 이 경우 cross-process, 즉 다른 메모리 공간에 있는 함수 호출을 투명하게 처리하기 위해서 각각의 프로세스에 구현되어 있는 proxy 와 stub들의 상호작용이 요구된다. proxy와 stub은 서버의 유형이 DLL 서버이든 EXE 서버이든 서버의 유형과 관계없이 동일한 방법으로 COM 객체의 서비스를 사용할 수 있으며, EXE 서버의 경우에도 로컬 서버이든 리모트 서버이든 서버의 위치와 관계없이 동일한 방법으로 서비스들을 사용할 수 있게 한다. 각 COM 객체는 서버 외부로, 즉 클라이언트에게로 하나 이상의 인터페이스를 사용하여 연결한다. 인터페이스는 COM 객체가 자신의 모습은 감추면서 동시에 자신이 제공하는 서비스는 클라이언트에게 노출시키고 싶을 때, 사용하는 도구이다.

클라이언트가 COM 객체를 사용하기 위해서는 우선 해당 컴포넌트가 클라이언트 시스템의 시스템 레지스트리에 반드시 등록되어 있어야 한다. 그런 후 먼저 사용할 COM 객체를 생성해야 하며, 다음으로 그 COM 객체에 연결되어 있는 사용하고자 하는 인터페이스의 주소를 조사해야 한다.[2]

COM이 제공하는 서비스의 종류로는 COM 객체가 가지고 있는 자료들을 클라이언트가 읽거나(Get) 쓸 수(Put) 있도록 해주는 속성(Property)이라는 서비스와 COM 객체의 동작을 유발시키는 메소드(Method)라는 서비스 등이 있다. 그러나 각 서비스는 반드시 특정 인터페이스에 소속되어 있어야 하며 이를 이용하여 COM은 외부와 통신을 하게 되며 서비스를 제공한다.[1]

V. 구현

5.1 PCI 확장카드 설계

PCI BUS 구조에서 카드에 대한 정보는 컴퓨터 내부에 있는 Bios에서 해당 카드의 정보를 읽어들여 카드가 필요로 하는 I/O Address, Memory Address, Interrupt Number 등을 충돌이 일어나지 않게 할당하여 준다. 이를 위하여 Serial EEPROM을 사용하였다. 그리고 PCI Interface chip은 PLX 9050-1을 사용하였으며 이 chip이 제공하는 local bus를 이용하여 Digital/Analog Data를 각각 16bit, 12bit로 전송하도록 하였다.

하드웨어 보드는 크게 Digital Input/Output, Analog Input/Output와 PCI Interface 부분으로 나누어진다. PCI BUS와 연결되는 Interface 부분에는 Interface chip과 데이터의 방향을 조절하는 Transceiver, Digital/Analog Data 인지를 결정하는 PAL, 카드에 대한 정보가 들어가 있는 EEPROM이 있다. 여기서는 Local 쪽의 데이터를 컴퓨터 내부로 보내고 받는 기능과 Digital, Analog Data를 구별하고 데이터의 전송방향을 결정하는 역할을 한다. PCI Interface를 그림 5에 나타내었다.

5.2 Driver 소스 코드에서 필요한 함수 분석

WinDriver에서는 Driver Wizard를 이용하여 진단 프로그램을 생성한다. 이 프로그램은 디바이스의 동작이 개발자의 의도에 맞게 정확히 작동하는지를 검사할 수 있다. 이 프로그램 소스 내에는 xxx_lib.c와 xxx_lib.h이 있다. 이 파일들 내에는 디바이스를 제어하는 함수들이 정의되어 있다.

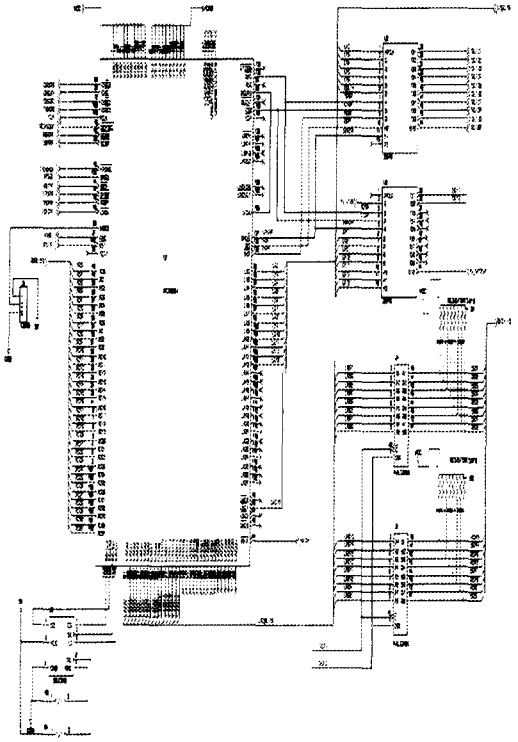

그림 5. PCI Interface

COM 프로그램에서 디바이스를 제어하기 위해서는 먼저 WinDriver 프로그램 소스 코드 내에 Device Open/Close에 관련된 함수가 필요로 한다. 그리고 데이터의 입출력을 위해 Device Read/Write 관련 함수가 필요하다. WinDriver에서 제공되는 소스들 내에 중요한 함수를 정리하면 표 1과 같다.[4]

표 17 드라이버 함수

동작	관련 함수
디바이스 열기	xxx_Open(…)
디바이스 닫기	xxx_Close(…)
디바이스 읽기	xxx_ReadByte(…) xxx_ReadWord(…) xxx_ReadDword(…)
디바이스 쓰기	xxx_WriteByte(…) xxx_WriteWord(…) xxx_WriteDword(…)

5.3 COM 설계 및 서버 프로그램 구현

COM을 설계하는 데 가장 먼저 해야 될 사항은 인터페이스 설정이다. 인터페이스 설정에서 고려해야 할 사항은 사용하고자 하는 메소드와 속성들의 관계가 종속적인지를 파악한다. 만약 메소드와 속성 중 독립적인 구성이 있는 경우 그것을 포함하기 위한 인터페이스를 새로 만들어 내야 한다. 본 연구에서 제어할 디바이스는 하나이므로 모든 메소드와 속성은 하나의 인터페이스에 포함될 수 있다. 다음 작업은 인터페이스 내부에 들어갈 메소드와 속성을 정의하는 것이다. 인터페이스를 디바이스로 가정하고 드라이버의 동작과 동일하게 하기 위해서는 디바이스 열기/닫기가 포함되어야 한다. 그리고 데이터의 입출력을 위해 디바이스 읽기/쓰기도 포함되어야 한다. 이것을 바탕으로 인터페이스 내부에 디바이스를 제어하는데 필요한 변수와 메소드를 작성하였다. COM 메소드를 정리하면 표 2와 같다.

표 18 메소드

동작	명칭
디바이스 열기	DeviceOpen
디바이스 닫기	DeviceClose
디바이스 읽기	DeviceReadWORD
디바이스 쓰기	DeviceWriteWORD

앞서 분석된 드라이버 중요 함수와 COM의 메소드는 동일한 형태로 되어 있다. 그래서 실제 구현한 COM 서버 프로그램 내에 각 동작에 해당되는 드라이버 함수를 포함시킴으로서 인터페이스의 메소드를 드라이버와 연결시킬 수 있다.

5.4 COM 클라이언트 프로그램 구현

클라이언트 프로그램을 작성할 경우에는 COM 서버 프로그램 템플릿 파일을 클라이언트 프로그램 내에 포함시킨다. 포함시키는 방법은 소스 내의 선언부에 #import "COM서버프로그램.tlb"을 포함시켜 준다. COM과 연결할 인터페이스 변수를 선언하고 소스 내에서 인터페이스 연결하고 사용한 후 연결을 끊으면 된다. 제작된 디바이스의 기능은 Digital/Analog 신호의 입력과 출력이다. Digital 입출력은 하나의 I/O map 주소를 통해서 이루어진다. 그리고 Analog 입력은 16개의 채널을 통해 들어오고 출력은 2개의 채널을 통해 나간다. Digital 입력의 경우와 출력의 경우 사용되는 I/O map 주소가 다르다. 또한 Analog의 경우 입력 채널과 출력 채널의 주소가 다르다. 그렇기 때문에 각각의 경우에 맞게 클라이언트 프로그램을 맞추어 각 입출력 메소드의 파라미터를 각 해당 I/O map 주소로 맞추었다.

Digital 신호의 경우는 입력 데이터와 출력 데이터 모두 16 bit이며 Analog 입출력 데이터의 크기는 12 bit이므로 프로그램에 입력하는 부분과 출력되는 부분을 16진수 데이터 형태로 출력하였다. Digital/Analog의 경우 모두 데이터의 처리가 연속적이므로 타이머를 이용하여 각 시간마다 입출력을 실행하였다. 프로그램의 개발환경은 Window 95 이상이고 개발언어는 비주얼 C++ 6.0 이다. 그림 6은 실제 구현한 프로그램 예이다.

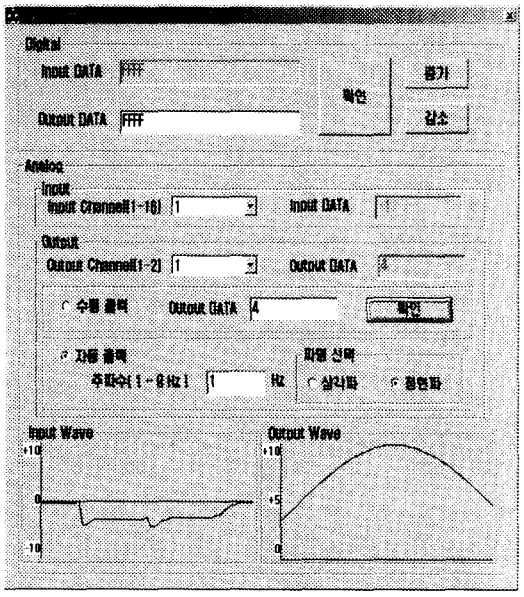

그림 6. Digital/Analog 입출력 제어프로그램

VI. 결 론

COM은 software 공학 기술 중에서 가장 경쟁력 있는 기술로 각광받고 있는 컴포넌트 기반 드라이브 인터페이스 개발 기법으로 시스템간의 호환성과 재사용성을 제공하며 높은 생산성과 고품질의 software 개발이 가능 할 것으로 기대된다. 현재 전세계적으로 컴포넌트 소프트웨어 시장이 팽창 할 것으로 예견하며 컴포넌트 개발이 활발히 추진 중이다. 현재 우리 나라도 소프트웨어 산업 육성에 의해 컴포넌트 산업의 활성화 노력은 IT산업 전반에 확산될 것이며 급변하는 산업환경에 신속하게 대처하여 경쟁력을 높여 줄 것이다.

PCI BUS상에서 동작할 수 있는 D/I, D/O 디바이스를 개발하고 COM 기술을 적용한 디바이스를 제어 할 수 있는 서버와 클라이언트 프로그램을 개발하였다. 그래서 일반 사용자들이 아주 손쉽게 입출력 디바이스를 사용하여 산업현장과 컴퓨터를 연결하여 자동화 할 수 있으며, 기존에 공장 자동화 프로그램에 종속되어 있으므로 고가로 구입하였던 입출력 디바이스를 상대적인 저가로 구입하여 사용가능 하리라 여겨진다.

앞으로 연구 방향은 다음과 같다. 첫째 본 시스템을 바탕으로 다양한 컴포넌트를 개발하는 것이고 둘째 인터넷을 기반으로 웹에서 이러한 서비스를 제공할 수 있도록 시스템을 확장하는 것이다.

참고 문헌

[1] 강익태, Visual C++ 6.0 ATL 컴포넌트 제작, 인솔미디어, 2000.
[2] Grimes, Stockton, Reilly& Templeman, Beginning ALT COM Programming Wrox, 1999
[3] 전병선, Microsoft C++6.0 ATL COM Programming, 삼양출판사, 2000
[4] WinDriver Developer's Guide KRF Tech, KRF Tech
[5] "PCI Local Bus Specification Revision 2.1", Portland : PCI Special interest group, 1995.
[6] 송호용, "PCI의 효율을 고려한 시스템 버스 설계에 관한 연구", 서울대학교 대학원석사 논문, 1997.
[7] 동역메카트로닉스연구소 기술정보실 편서, PCI 버스 해설과 인터페이스 카드 설계, 국제테크노정보연구소, 2000.
[8] Shanley, T. Anderson& D. Mindshare, Inc (Cor), "PCI System Architecture", Addison-Wesley Pub Co (C), 1999.