MFC 와 객체지향 프로그래밍을 이용한 FPD Stocker 의 개발

김성원, 백두산, 김석동, 김우성
호서대학교 컴퓨터공학과
E-mail: wissen@dreamwiz.com

Development of a FPD Stocker with MFC and Object-Oriented Programming

Sung-Won Kim, Doo-San Baek, Suk-Dong Kim, Woo-Sung Kim
Dept. of Computer Engineering, Hoseo University

요 약

FPD Stocker System 은 LCD 를 포함한 평판 디스플레이 소자(FPD : Flat Panel Display)의 자동 자재 저장 반출 장치(FPD Automated Storage and Retrieval System)이다. FPD Stocker System 은 그 특성상 생산라인의 구조에 따라 보유 디바이스와 디바이스의 설정이 다양해 지는 특성이 있다. 본 논문은 새로운 디바이스의 추가가 쉽고, 그 디바이스의 설정이 용이한 구조의 FPD Stocker System 개발을 목적으로 한다. 이를 위하여 각 디바이스를 클래스로 구현하여 개별적인 스레드(Thread)로 Work Crew Threading Model 을 사용하여 동작 시켰다.

1. 서론

LCD(Liquid Crystal Display)를 포함한 평면 디스플레이 소자(FPD : Flat Panel Display)의 생산을 위해서는 여러 단계의 공정을 거쳐야 한다. 이러한 경우 하나의 공정이 끝나고 다음 공정으로 이동하기 전까지, 다음 공정을 진행할 라인으로 이동하고 나서도 공정 처리 장비에 들어가기 전까지 FPD 를 임시로 저장해 놓을 공간이 필요하다. FPD Stocker 는 FPD 를 수납한 FPD Cassette 를 저장하여 생산 자동화를 이룰 수 있도록 한다.

기본적으로 FPD Stocker System 은 LCD 를 포함한 평판 디스플레이 소자(FPD)의 자동 자재 저장 반출 장치이다. 즉, 다수의 평판 디스플레이 소자를 포함하는 Cassette 단위의 자재를 입출력 포트와 로봇시스템이 자동 반송하여 임의의 위치에 있는 선반에 저장, 또는 반출하는 장치인 것이다.

저장된 자재의 이력은 데이터베이스에 저장되어 Stocker 에 장착된 LCD 모니터를 통해 관리할 수 있으며 SECS/GEM 등의 표준화된 통신 프로토콜을 통해 전체 공정을 관리하는 Host 로 보내진다.

FPD Stocker System 은 생산라인의 규모나 생산 라인에 투입되어 있는 생산장비의 종류 나 방식등에 의해 다양한 설정을 갖을 수 있다. 예를 들면 ID 를 읽기 위한 디바이스가 Bar Code Reader 가 될 수도 있고, 전기적인 타입, 적외선 타입, RF(Radio Frequency) 타입의 Code Reader 가 사용될 수도 있다. 필요에 따라 추가되는 디바이스도 있고 장착이 안되거나 다른 종류의 디바이스가 장착되는 경우도 있다. 본 시스템에서는 Windows Registry 를 이용한 Configuration 으로 새로운 디바이스의 추가나 설정이 매우 용이하도록 하였다.

국내 반도체 제조회사에서 연구 사용하는 FPD 용 Stocker 는 주로 일본 제품으로 장비의 본체인 운영 소프트웨어를 수입해서 사용하는 중이다. Stocker 의 운영 소프트웨어는 생산 자동화에 중요한 부분이므로 국산화 연구가 절실히 필요하다.

2. 전체 시스템 구조

Stocker 용 운영 소프트웨어는 Stocker Engine 과 Stocker GUI 의 두개의 주요부분으로 구성되어 있다.

그리고 시스템의 특성에 따라 환경을 설정할 수 있는 Configuration(Windows Registry)과 자재의 이력 관리와 사용자 권한을 저장하기 위한 Database 를 가지고 있다.

분리되어 있는 Engine 과 GUI 두 프로그램은 TCP/IP 등의 네트워크를 통해 구성된 Message Bus 로 통신한다.

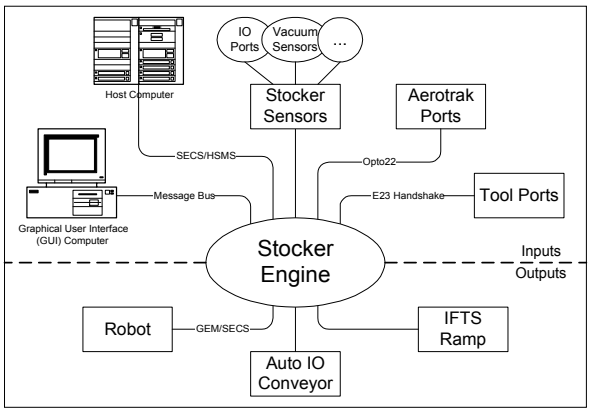

그림 1 FPD Stocker System 구조

FPD Stocker Hardware 는 FPD Cassette 를 저장하는 창고이다. 겉에서 보기에 FPD Cassette 를 넣거나 꺼낼 수 있는 Port 라는 입구가 저장/반출 용으로 각 하나씩 있다. Port 에는 Bar Code Reader, Light Curtain, 각종 센서 등의 부가적인 디바이스가 장착되어 있다.

FPD Cassette 를 저장할 경우 Input Port 로 입력된 FPD Cassette 는 Robot 에 의해 이동되어 쉘프(Shelf)라는 장소에 보관된다. 반대로 FPD Cassette 를 반출할 경우 사용자나 호스트의 명령에 의해 해당하는 FPD Cassette 가 쉘프로 부 터 Robot 에 의해 이동되어 Output Port 를 통해 반출된다.

2.1 시스템 사양

System Specifications	
Operating system:	Microsoft Windows NT V4.0
Development platform:	Microsoft Visual C++ V6.0
Device driver support:	Blue Water System WinRT V1.0
Database platform:	Microsoft Data Access Objects (DAO) V3.0
(optional) Touch screen support:	Microtouch touch screen V4.0
(optional) Uninterruptible power supply support:	Deltec LANSafe V4.7
Hardware platform:	PC compatible computer with 200Mhz Pentium processor (minimum) 64MB RAM (minimum) 2.5Gigabyte hard disk drive (minimum)
Network platform:	Ethernet Network card with both RJ-45 and BNC connectors to support either 10-BaseT or 10-Base2 cabling.
Display:	Liquid Crystal Display capable of a minimum of 640x480 resolution.
Pointing device:	Touch screen and/or trackball
(Optional) Un-interruptible power supply (UPS):	Capable of a minimum of 400kVA and supporting NT interrupt signaling.

그림 2 시스템 사양

2.2 Stocker Engine

Stocker Engine 은 Stocker 운영체제의 핵심으로 모든 센서들의 입/출력을 관장하고, Host 또는 사용자로부터의 요구를 수신하는 등의 작업을 우선순위와 장비 사용가능 여부에 따라 스케줄 한다. 그리고 시스템의 설정에 따라 연결되는 Robot, Automatic I/O, Port 등에게 적절한 명령을 생성 전달하고 데이터베이스를 유지 및 갱신한다.

Stocker Engine 은 Command Queue 를 통해 Stocker GUI 나 Host 에서 보내진 명령을 순차적으로 처리한다.

FPD Stocker System 에는 FPD Cassette 를 입/출력 하는 Port, FPD Cassette 의 ID 를 읽는 ID Reader, FPD Cassette 가 Port 의 제자리에 올려 졌는지 등을 검사하는 Through beam 등의 각종 센서, Port 에 올바르지 않게 Cassette 나 사람의 손등이 들어오는 것을 감지하는 Light curtain, FPD Cassette 를 옮기는 Robot 등 다양한 디바이스가 장착되어 있고, 각 디바이스는 클래스로 그 기능이 구현되었다.

각 디바이스는 시스템에 다양한 종류가 다양한 형태로 다양한 조합으로 부착되어 있고, 각 디바이스의 설정과 디바이스간의 연결 방식 그리고 그 세부 설정 등이 Windows Registry 에 기록되어 진다.

FPD Stocker 는 초기화 시에 Windows Registry 를 읽어 디바이스 설정 상태를 얻어오고 해당하는 디바이스를 사용하기 위한 클래스를 인스턴스(Instance) 화 하여 개별적인 스레드(Thread)로 구동 시킨다.

구동 된 스레드는 Work Crew Model 방식으로 동작하며 다른 스레드와 상호 협력하면서 작업을 진행하게 되고 이를 위하여 Windows event 개체를 이용한 Callback system 으로 스레드간 의사 소통을 하게 된다.[5]

각 스레드는 Callback 을 통해 메시지를 주고 받으며, 서로간에 자원을 예약(Reserve)하고 자원을 사용하는 방법으로 작업을 진행시키게 된다.

각 Windows Registry 에는 디바이스 이름과 해당 디바이스를 구동하는 클래스의 이름이 들어가게 된다.

FPD Stocker 는 초기화 시에 문자열로 전달되는 클래스 이름을 가지고 동적으로 클래스를 인스턴스 화 해야 한다. 이를 위하여 RTTI(Run-Time Type Information)을 사용한다.

RTTI 는 프로그램이 실행 중에 개체의 형을 알아내기 위한 메커니즘이다[4]. 그림 3 에 C++에 서 지원하는 RTTI 메커니즘을 소개하였다.

- The **dynamic_cast** operator
  - Used for conversion of polymorphic types.
- The **typeid** operator
  - Used for identifying the exact type of and object.
- The **type_info** class
  - Used to hold the type information returned by the typeid operator.

그림 3 RTTI (Run-Time Type Information)

- 각 스레드 모델별 특성. 본 논문에서는 Work Crew Model 을 사용한다.
  - Boss/Walker Model
    - Master thread 가 Worker thread 에게 작업을 할당
    - 보스 워커 모델에서는 서로 상호 작용할 필요가 있다.
  - Work Crew Model
    - 여러 개의 thread 가 하나의 작업을 처리 하도록 협력
    - OSF/DCE Guide 에서는 위크 크루 모델을 집을 청소하는 사람들에 비유하고 있다.[4]
  - Pipelining Model
    - 하나의 작업을 흐름에 따라 조각 내어 각각의 여러 개의 thread 에서 차례로 처리
    - 여러 개의 결과를 만들기 위해 여러 개의 thread 에서 병렬로 처리한다.
    - OSF/DCE Guide 에서는 파이프라이닝의 예로 공장의 조립라인을 예를 들고 있다.[4]

- Callback
  - Callback 은 기본적으로 함수 포인터 형태이다.
  - 예를 들어 Class A 에서 Class B 로부터 무언가를 기다린다면, Class A 는 Class B 에게 Callback 개체를 전달하고 Class B 는 전달받은 Callback 개체를 이용하여 Class A 에게 메시지를 보낼 수 있다. 그림 4 참조

2.3 Stocker GUI

Stocker GUI 는 Stocker Engine 과 작동자(Operator)의

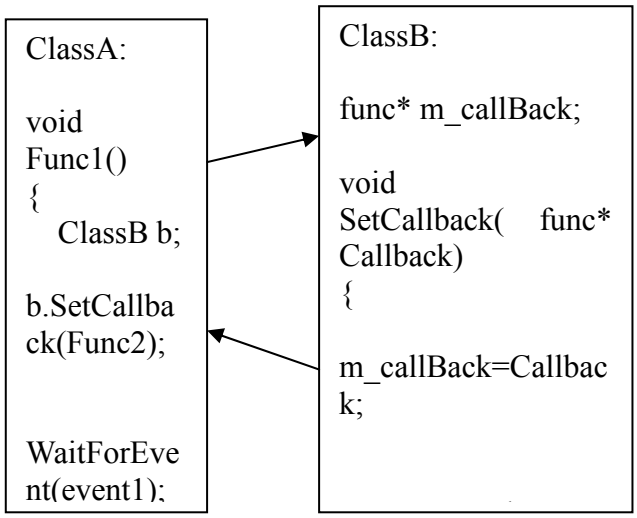

그림 4 Callback

인터페이스 역할을 담당하며 작동자가 Stocker 에 명령을 내릴 수 있도록 한다. 작동자의 실제 위치와 사용권한에 따라 GUI 는 어떠한 특정 명령을 보내거나 특정 수준의 에러를 지울 수 있도록 해 준다.

Stocker Engine 과 Stocker GUI 의 연결 방식에 따라 Local / Remote / Mixed 로 나눌 수 있으며 내용은 다음과 같다.

- Local Mode : Stocker Engine 은 단지 그래픽 사용자 인터페이스를 통해서 명령을 받아들인다. 사용자는 Cassette 를 저장/검색 할 수 있다.
- Remote Mode : Stocker Engine 은 외부 호스트 컴퓨터로부터 명령을 받는다. 사용자는 Stocker GUI 를 통해 Cassette 를 저장/검색 할 수 없다.
- Mixed Mode : Stocker Engine 은 Stocker GUI 와 호스트 컴퓨터로부터 명령을 받아들인다. 호스트 명령들은 GUI 통해 취소될 수 없다. 도한 GUI 명령들은 호스트에 의해 취소될 수 없다.

GUI 는 4 개의 사용자 액세스레벨(Access Level)을 가지고 있어 권한에 따라 기능과 GUI 화면에 제한을 두고 있기 때문에 권한이 없는 사용자로부터의 장비 보호가 가능하다.

Stocker GUI 는 사용자의 보안등급에 따라 아래와 같은 4 등급의 액세스 권한이 있다.

- 주변(Ambient) : 가장 낮은 액세스 권한으로 사용자가 Stocker GUI 에 로그인하여 Stocker 작동을 모니터 할 수 있지만 이 등급의 사용자는 다른 어떤 조작도 수행할 수 없다.
- 작동자(Operator) : 이 권한은 사용자가 FPD Cassette 를 시스템에 저장하거나 반출 하도록 한다.
- 감독자(Supervisor) : 이 권한은 사용자가 모든 작동자 기능을 실행하고 Stocker 내부에서 쉘프와 쉘프 간의 자재를 이동하며, 사용자나 시

스템에 위험하지 않은 오류(Event)를 복구할 수 있도록 허용한다.

- 고급사용자(Superuser) : 최고 권한의 액세스로서 감독자 권한에서 할 수 있는 모든 기능을 수행하거나 실행할 수 있으며 사용자가 GUI 화면 구성 요소들을 수정하거나 종료할 수 있다. 또한 Windows NT 운영 체제 데스크 탑, 시스템 구성 데이터 및 관리자 등급 암호 입력 대화 상자에 액세스할 수 있다.

GUI 는 네트워크를 통해 Stocker Engine 과 연결되고 Message Bus 를 사용하여 통신한다. 필요에 따라서 Stocker Engine 한대에 여러 대의 Stocker GUI 가 연결될 수 있다. 이 때 Stocker GUI 는 Stocker Engine 의 데이터베이스를 공유하여 사용하도록 설계되었다.

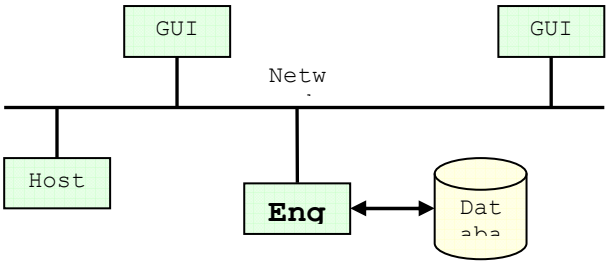

그림 5 Stocker Engine 과 Stocker GUI

2.4 Software Configuration

System 의 환경 설정은 Windows Registry 를 통해서 이루어 진다. Registry 의 키에 사용할 디바이스와 속성을 편집함으로써 Stocker Engine 이 초기화 될 때 Windows Registry 를 읽어와서 해당 장치를 초기화 한다.

Stocker Engine 은 기능 확장의 유연성과 환경설정을 하기 편리한 구조로 디자인 되었다. 이것은 가능한 한 OOP 적으로 설계했기 때문에 가능하다.

FPD Stocker System 에는 FPD Cassette 의 입출력을 위한 Port, FPD Cassette 를 이동하기 위한 Robot, 각종 Sensor, FPD Cassette 의 ID 를 읽는 ID Reader 등 다양한 종류의 디바이스가 장착 되게 된다.

Windows Registry 에는 디바이스와 디바이스를 사용하기 위한 클래스의 이름이 문자열로 저장된다. FPD Stocker 는 초기화 될 때 디바이스를 사용하기 위한 클래스 이름을 읽어 실행 중에 개체화 시킨다.

2.5 Information Storage - Database

Stocker Engine 은 Stocker.MDB 와 StockerGUI.MDB 2 개의 데이터베이스를 가지고 있다. Stocker Engine 에 의해 관리되는 Stocker.MDB 는 7 개의 테이블을 가지고 있으며, 주요 기능은 환경 설정과 자자의 이력 관리 부분이다.

StockerGUI.MDB 는 사용자의 액세스 등급과 화면 디스플레이에 관한 정보, Command Queue 의 내용을 복사하여 보존하는 역할을 한다.

Database 에 연결하기 위해 MFC DAO 클래스(Version 3.5)를 사용하여 마이크로소프트 액세스에서 생성된 MDB 액세스 데이터베이스 파일에 접근할 수 있으며 하나의 테이블 또는 질의문을 위해 CDaoRecordset 클래스에서 상속 받아 객체를 만들어 필요한 질의를 하게 된다.

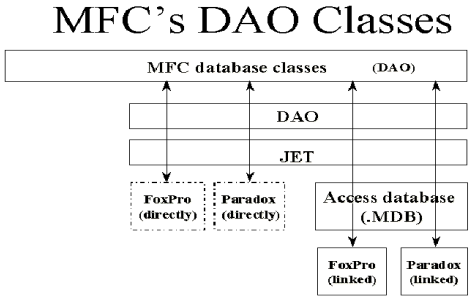

그림 6 MFC DAO Classes

2.6 Message Bus

Stocker Engine 과 Stocker GUI 는 Message Bus 를 통해서 메시지를 주고 받는다.

이 메시지 버스는 TCP/IP 를 이용하며, remote GUI 와 local GUI 모두 같은 엔진과 통신한다. 각각의 메시지는 메시지의 종류를 나타내는 고유의 ID 를 가지고 있다.

각각의 프로그램은 Message Bus 를 사용하기 위해 별개의 스레드를 이용하며, 이 스레드는 메시지를 받아서 Windows 사용자 정의 메시지로 다시 조합하여 프로그램에 전달한다.

2.7 호스트(Host) 통신

호스트 시스템은 원격 감시와 반도체 제조 장비간의 데이터 교환을 지원하는 시스템으로 RC-232C 통신 채널이나 Ethernet 을 사용하는 통신 환경이 제공된다. 이중 주로 사용되는 RS-232C 통신 채널을 사용한 호스트 시스템은 반도체 제조 장비의 데이터를 송수신하여 원격 감시를 수행하며 이를 위해 SECS 프로토콜이 통신 규약으로 사용된다[1]. SECS 프로토콜은 전속 규정에 의거하여 제한된 크기의 메시지 블록 단위로 데이터가 처리된다. 따라서, 호스트 통신 인터페이스 모듈은 SECS 프로토콜이 제시하는 계층 규약을 만족하도록 설계되어야 한다[2][3]. SECS 프로토콜 계층은 아래 그림과 같이 SECS-I, SECS-II 로 나뉜다.

SECS-I 계층은 프로토콜의 메시지 전송 부분으로 RS-232C 통신 표준을 따르는 하드웨어 인터페이스를 담당하는 하위 영역과 데이터가 전송되는 방법을 위한 기능인 전송 제어, 메시지와 데이터 블록의 트랜잭션을 담당하는 상위 영역으로 구성된다.

호스트 시스템의 통신 모듈을 위한 SECS-I 계층에서의 요구 사항은 RS-232 프로토콜을 이용한 외부 인터페이스, 메시지를 통해 블록의 형태를 정의하고 관리하는 메시지 관리 기능, 데이터 블록의 형식을 정의하고 전송되는 블록의 에러를 처리 할 수 있는 블록 송수신 기능, 하나 이상으로 정의된 메시지 전송 규정 즉, 트랜잭션을 관리하는 기능이 요구된다.

SECS-II 계층은 프로토콜을 구성하고 있는 전체 메시지의 내용을 정의하고 정의된 메시지 구조체의 내용과 형식에 따라 데이터의 변환을 통해 응용 프로그램으로 전달된다. 모든 메시지의 내용은 스트림(stream)과 기능(function)으로 구성되며 이를 위한 요구 사항으로 데이터 항목 생성, 메시지 구조체 생성, 메시지 전송 규정 관리 기능이 요구된다.

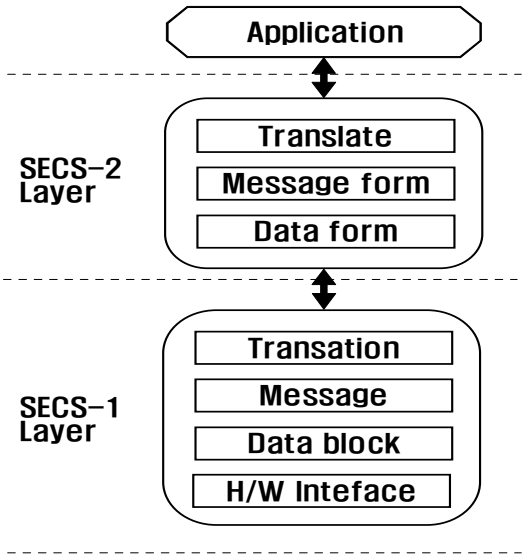

그림 7 SECS

2.8 Safety Systems

예기치 않은 사고에 의해 인명의 상해나 FPD Cassette 의 파손, FPD Stocker System 의 파손을 막기 위한 기능이다.

- Light curtains : Robot 이 동작중인 경우 Port 를 통해 사람의 손이나 FPD Cassette 가 들어올 경우 사고가 발생할 수 있다. Light curtain 은 Port 의 입구를 감시하여 부적합하게 Port 로 입력되는 경우를 감시한다.
- Access service door interlocks : FPD Stocker 가 동작중인 동안 FPD Stocker 안으로 사람이 들어가지 못하도록 보호한다.
- Light stack : 현재 시스템 상태를 램프를 통해 알린다.
- Sonic alarm : 에러 발생이나 처리 결과 등을 사람이 들을 수 있는 가청 주파수로 알린다.
- Cassette inventory scanner : 이미 쉘프에 Cassette 가 들어 있을 경우 해당 쉘프에 또 다른 Cassette 를 넣으려고 할 경우 사고가 발생하게 된다. 쉘프 에는 장비가 꺼져있는 경우등에 작업자등에 의해 Cassette 가 들어갈 수도 있으므로 FPD Stocker 는 쉘프 로의 동작중에 쉘프의 상태를 확인하게 된다.
- EMO, robot pause buttons : 사고 발생시나 사고 예견시 작업자에 의해 시스템을 정지 시킬 때 사용한다.

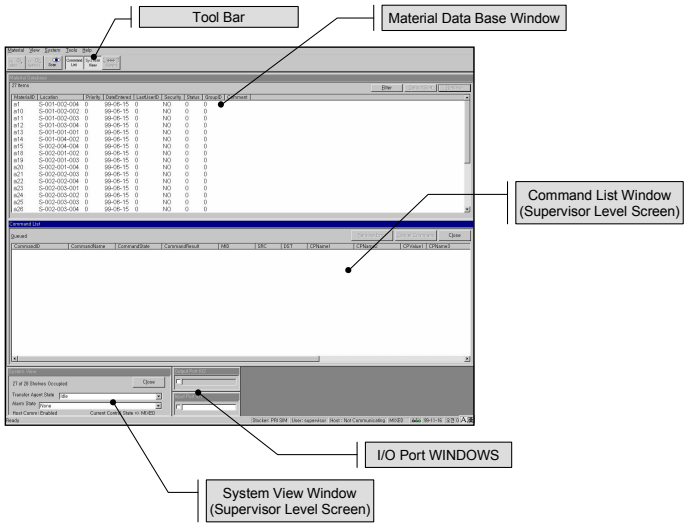

그림 8 Stocker GUI

3. 결론

평판 디스플레이 소자의 저장/반출을 위해 개발된 본 시스템은 생산라인의 다양한 요구를 수용할 수 있도록 개발되었다. Windows NT 환경에서 Registry 에 따라 시스템이 구성되고, 시스템을 구성한 각 디바이스는 독립적인 스레드에서 독립적으로 실행된다. 각 스레드는 CALLBACK System 을 통해 서로간에 협력하여 작업을 수행해 나간다.

향후 보다 간단한 하드웨어와 운영 소프트웨어로 시스템의 개발하여 생산 단가를 낮출 필요가 있다.

참고문헌

[1] SEMI(Semiconductor Equipment and Materials International), STANDARDS PUBLICATIONS, http://www.semi.org.

[2] 박성식, 최현일, 최용엽, “Windows NT 에서 반도체 제조 장비와 호스트간 메시지 전송을 위한 통신 시스템의 설계”, 한국통신학회 ‘97 하계종합학술발표 논문집, pp.959-962.

[3] 김평진, 오삼권, “SECS(SEMI Equipment Communication Standard) 시스템의 요구사항 분석”, 한국정보처리학회 ’98 춘계학술발표 논문집, 6 권 1 호, pp.1141-1144.

[4] Microsoft, MSDN Library

[5] Silberschatz, Galvin “Operating System Concepts,” Addison Wesley, 1994.