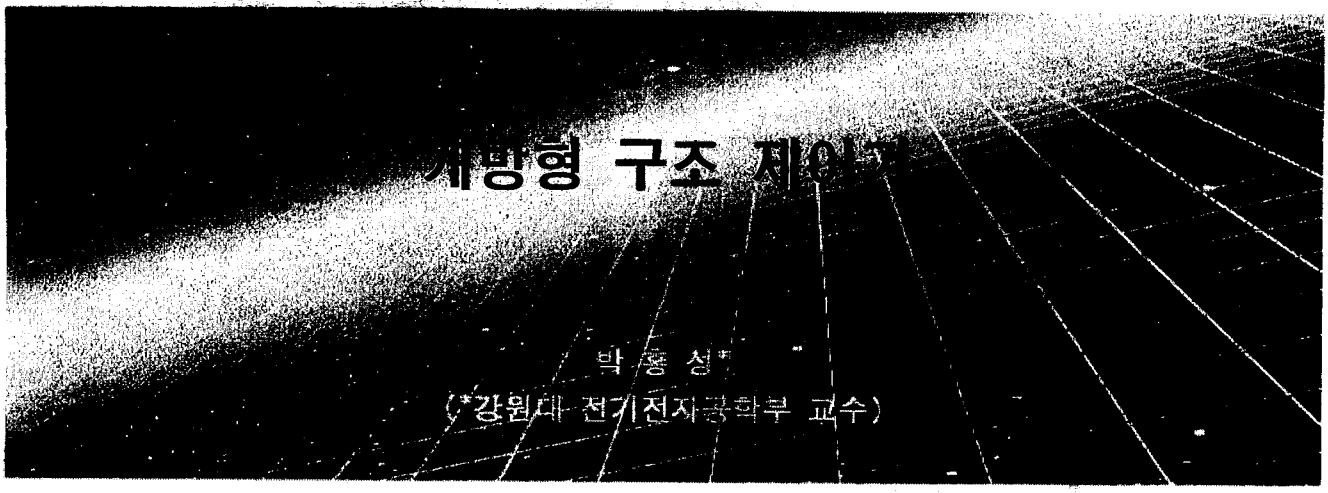


개방형 구조 제어기

박 홍 성*
(*강원대 전기전자공학부 교수)

1. 서 론

현재 그리고 미래에서도 생산 경쟁력을 유지하기 위해서는 생산 공정을 빠른 시간 내에 적절한 가격으로 향상시킴으로써 시장 요구에 대응해야 한다. 이러한 요구를 만족시키고 위해 제품 생산자들은 보다 유연하고 민첩한 생산 시스템을 구축하고 있다. 특히 높은 경쟁력을 얻기 위해서는 보다 향상된 제어 기술을 공정에 적용하는 것이 매우 중요하다. 현재 사용되는 고전적인 생산/제어 시스템들은 새로운 제어 기술들의 통합 및 갱신을 할 수 있도록 설계되어 있지 않다. 따라서 생산 공정과 장비는 새로운 공정 기술을 이용하고 제품 사양에 적용할 수 있도록 설계되어야 한다.

Frost & Sullvian의 자료에 따르면 1996년도의 CNC의 세계 시장규모는 1.3억불이며, Automation Research Co.의 자료에 따르면 1997년 PLC의 세계 시장 규모는 53억불이다. 미국의 로봇 시장은 1998년에 7억불로 예측하고 있으며, DCS의 세계시장규모[1]는 1998년에 25억불로 예측하고 있다. Dataquest 자료에 따르면 1993년 PC 시장 규모는 66억불, 데이터 통신 시장은 14억불이다. 이외에 컴퓨팅 및 정보통신 시장은 다양하며 1998년을 기준으로 보면 PC 시장과 정보통신 시장 규모는 엄청난 것이다. 이러한 것을 비교해 보면 연산 능력과 데이터 통신의 진보는 산업용 제어기 벤더보다는 이러한 시장에 적극적인 회사에 의해 좌우된다는 것을 알 수 있다. 즉, 정보통신 및 컴퓨팅 분야에서 사용되는 기술들이 산업 분야의 제어기 구조를 바꿀 수 있다는 것을 알 수 있다.

이러한 경향을 적극적으로 수용하기 위해서는 새로운 형태의 생산 시스템이 필요하게 되었다. 이러한 필요성은 새로운 형태의 생산 시스템, 재구성가능생산 시스템(reconfigurable manufacturing system)을 제안하게 되었다. 이러한 시스템은 빠르고 신뢰성 있게 재배열할 수 있는 기본 공정 모듈들을 사용하여 만들어진다. 재구성가능 시스템을 구현하여 동작시키기 위해서는 시스템 레벨, 기계(하드웨어) 레벨, 제어 레벨에서의 구조가 새로운 모듈과 기능들의 추가 및 통합을 가능하게 하기 위한 개방형이어야 한다. 다시 말하자면, 재구성가능 생산 시스템에 가장 중요한 부분은 새로운 세대의 제어 시스템인 개방형 구조 제어기이다.

새로운 기능이라 하면 생산 기기에 새로운 툴 매거진을 구축하는 것에서부터 부품의 크기를 감시하거나 정밀도를 높이기 위해 기계 및 플랜트에 보정 신호를 보내는 센서를 붙이는 것, 필드버스를 연결하는 것까지 매우 다양하다.

그러나, 시장에서 개방형 제어기의 채택은 매우 느리다. 이에 대한 주된 이유는 규모가 큰 제어 벤더들의 저항이며, 이는 개방형구조 제어기의 개발이 그들의 사업적인 흥미(즉, 이윤 보장)를 채워주지 않았기 때문이다. 그들은 기존 모델을 갱신하는 것보다 새로운 제어기를 판매하는 것을 선호하며, 시장을 완전히 지배하기를 원하기 때문이다. 그럼에도 불구하고 미국에서는 GM과 미 공군 등에서 개방형 구조 제어기에 대해 끊임없이 요구함으로써 개방형 제어기 방향으로 변화시키고 있다. 현재는 개방형 구조제어기는 성숙되자마자 모든 사용자와 머신 빌더들이 채택하기를 원하는 기술이 되었다.

따라서 많은 국가에서 개방형 구조 제어기와 관련된 기술을 개발하고 있다. 본 고에서는 세계 각국에서 개발되고 있는 개방형구조 제어기(예, OSACA[2], OMAC[3-6], OSEC[7,8], OPC[9])의 내용을 살펴보고 국내의 개발 방향을 논의할 것이다.

2장에서는 개방형구조 제어기의 정의와 수준에 대해 살펴보며, 3장에서는 개방형 구조 제어기의 장단점을 살펴보며 4장에서는 각종 개방형구조 제어기 과제의 내용과 그들을 비교하며 5장에서는 국내의 개발 방향에 대해 논의하고 결론을 맺겠다.

2. 개방형 구조 제어기의 정의

개방형 구조 제어 시스템을 정의하기 위해서는 다음과 같은 IEEE 1003.0(1990)의 정의를 기본으로 한다.

"An open system provides capabilities that enable properly implemented application to run on a variety of platforms from multiple vendors, interoperate with other systems applications, and present a consistent style of interaction with the user."

이러한 정의를 기반으로 OSACA는 개방형 제어 시스템을 다음과 같이 정의하였다.

"An open control system consists of a set of logically discrete components. The interfaces between these components and between the components and the implementation platforms are well defined such that a meaningful combination of components from different vendors can cooperate with each other form a complete and correctly functioning control that runs on a variety of platforms and presents a consistent interface to the human users as well as other automation systems."

이러한 정의에 있어서 다음의 2가지 중요한 면들,

- **사실상의 표준 환경의 사용과**
- **상용 S/W와 H/W 구성 요소들의 유용성**

을 내포하고 있다.

정의들에 있어서 내부 변수(internal variable)는 제어루프 내의 축 위치 에러 혹은 인터폴레이터에 대한 입력들과 같은 변수들을 의미하며, 이 변수들에 대한 액세스는 새로운 모듈과 알고리즘을 쉽게 통합시킬 수 있기 때문에 매우 중요하다.

이와 같이 다양한 정의가 있지만, 결과적으로 다음과 같은 요구조건들을 정의함으로써 개방형 구조 제어기를 정의하는 것이 알맞다.

- **개방성 :** 모듈 내부 변수에 대한 접근을 하게 하는 능력
- **모듈성 :** 관련된 함수들이 하나의 요소의 부분으로 간주될 수 있도록 묶을 수 있는 정도
- **상호운용성 :** 제어기의 2개이상의 구성요소가 정보를 교환하고 사용할 수 있는 능력
- **이식성 :** 응용 소프트웨어의 능력을 유지하면서 한 환경에서 다른 환경으로 전달될 수 있는 용이성
- **신축성(scalability) :** 응용 요구에 따라 시스템의 성능 혹은 규모를 감소하거나 증가시킬 수 있는 용이성
- **확장성 :** 제 3자가 오리지널 제어기에서 제공하지 않는 능력을 추가할 수 있는 용이성

이러한 정의에 따라 개발되었다 하더라도 개방형 제어기는 천차만별이다. 따라서 현재는 어떠한 단계에 있으면서 앞으로의 방향을 가늠하기 위해 개방형구조 제어기의 수준을 정의한다[10].

첫 번째 수준은 가상(pseudo) 혹은 반쪽 개방형제어기이다. 이러한 제어기는 고정된 서브시스템(동작 제어 유닛, 입출력 시스템, 사용자 인터페이스 등)을 사용하여 제작된 PC 기반 제어 시스템을 말한다. 서브시스템간의 통합은 독자적인 통합과 통신 표준을 통하여 가능하다.

두 번째 수준은 부분(piece-wise) 통합 제어기로 표현된다. 이러한 통합 제어기는 구성 요소가 기능적 모듈(하드웨어와 소프트웨어 쌍으로)로 분리된다. 시스템 통합자는 요구 조건을 만족시키는 서브시스템을 제어기에 자유롭게 결합시키지만, 제어기의 초기화/구성 및 모듈간의 통신 하부구조의 구현을 위해 어떠한 상용 툴도 제공되지 않는다.

세 번째 수준은 API의 공통 집합의 정의와 구성 툴 및 API에 합치하는 통신 하부구조의 유용성을 통하여 얻을 수 있다. 이 경우에 있어서는 과도한 엔지니어링 노력 없이 사용자가 제어 시스템을 재구성할 수 있다. 재구성가능성과 신축가능성은 실체화되어 가지만 하드웨어 표준에 좌우된다.

마지막 수준은 제어 시스템으로부터 모든 독자적인 하드웨어 요소들이 제거되며 제어기 백플레인에 연결되는 동작제어 및 이산 로직 제어 카드와 같은 특정 하드웨어 없이 소프트웨어 제어 모듈만을 동작시키는 일반 프로세서를 가진 소프트웨어 기반 제어기를 만든다.

현재 유용한 개방형 상용 제품은 두 번째 수준에 해당된다.

3. 개방형 구조 제어기의 장단점

개방형구조 제어기를 크게 나누면 벤더특정형과 벤더중립형으로 나눌 수 있다. 실제적인 의미에서는 개방형구조 제어기는 벤더중립형을 의미한다고 볼 수 있다. 그러나, 현재까지 만들어진 벤더중립형 제어기를 살펴보면, 많은 다른 표준이 만들어져 있기 때문에 벤더 중립 비호환 제어기를 만들 위험이 있다는 것을 염두에 두어야 한다.

벤더 특정 개방형 구조 제어기는 벤더에 의해 설계된 계측제어 기기와 알고리즘들의 통합을 위해 제어 벤더에 의해 설계된 개방형 시스템이다. 시스템 설계에 익숙한 최초 벤더만이 알고리즘을 추가하거나 변경시킴으로써 제어기를 재구성가능하게 한다. 어떤 특수한 경우에서만 특정 알고리즘을 벤더의 지침 하에서 사용자가 추가시킬 수 있다. 이러한 유형의 제어기를 제공하는 회사는 fanuc, Cicinnati Mialcron, Cimetrix, Hewlett Packard, Siemens 등이다.

벤더 중립형 개방형 구조 제어기는 제 3자에 의해 재구성이 가능하도록 계측제어 기기 들과 응용 소프트웨어들의 통합이 되도록 설계된 개방형 시스템이다. 시스템 구조가 공개적으로 알려져 있기 때문에 어떠한 벤더도 제어기에 새로운 계측제어 기기 들과 알고리즘들을 추가할 수 있고 변경할 수 있고 통합할 수 있다. 이러한 방법은 많은 장점을 가지고 있다.

- *특정 벤더에 의해 제한되지 않고 제어 시스템을 확장할 수 있다.*
- *사용자는 시장에 있는 최상의 알고리즘(예, 적응제어, 퍼지제어)과 기기 들을 사용할 수 있고 원 제어기 벤더에 의해 제공되는 것들을 사용하지 않아도 된다.*
- *사용자는 제어 벤더에 그들 독자 알고리즘을 공개함*

이 없이 특정 알고리즘을 통합할 수 있다.

- *같은 구조를 사용하는 산업용 제어기들이 점점 호환성 있게 되어감에 따라 오퍼레이터 훈련비용과 유지보수비용을 절감할 수 있다.*
- *사실상의 표준 필드버스를 통한 입출력의 통합이 용이하다.*
- *공정감시와 진단기능 등의 통합과 같은 다양한 부가기능이 제공된다.*

만족할 만한 벤더 중립형 구조를 가지기 위해서는 공개된 벤더 중립형 구조에 합치하는 개방형 구조 제어기를 만드는 것이 중요하다.

많은 장점에도 불구하고 개방형 구조 제어기의 확산을 막는 약간의 풀리지 않는 법률적 및 기술적 문제가 있다[11]. 제 3자에 의한 제어기의 변경 때문에 발생하는 손실에 대한 법률적 책임 문제는 개방형 구조제어기를 구현하는데 주된 방해물이다. 이러한 신뢰성과 안전성에 관련된 의문은 다음과 같다. 손실은 왜 발생했는가? 변경이 손실 발생의 직접적 원인인가 간접적 원인인가? 누가 생산성 손실 혹은 손해에 책임이 있는가? 변경을 한 제 3자인가 그것을 주문한 사용자인가 혹은 오리지널 제어기 벤더인가? 여기에는 어떠한 확정된 대답이 있을 수 없다. 따라서 사용자들은 법률적 책임이 제한되는 덜 모험적인 벤더 특정 개방형 구조 제어기를 선호할 수 있다.

이러한 법률적 문제 외에 벤더 중립형 개방형구조 제어기를 구현하는데 있어서 발생하는 기술적 문제를 간과하지 말아야 한다. CNC제어기는 주어진 샘플링 주기 내에 실시간으로 동작해야하기 때문에 제어기 재구성 단계에서 타이밍 조건을 맞추어야 한다. 따라서 계측제어 소프트웨어 모듈들을 특별한 응용에 맞도록 제작된 제어 시스템에 통합되도록 도와주기 위한 소프트웨어 툴이 개발되어야 한다. 내부 변수들을 직접 액세스하거나 모듈내의 알고리즘을 대체했을 때 제어기의 동작에 있어서 투명성과 신뢰성을 유지하는 것은 어렵다.

4. 각종 개방형 구조 제어기 과제

이 절에서는 현재 가장 많이 참조되고 있는 4개의 과제, OSACA, OMAC-TEAM, OSEC 및 OPC들에 대한 내용을 기능, 구조, API, 구성 툴 관점에서 살펴본다.

4.1 OSACA(Open System Architecture for Controls within Automation systems)[2]

1993년부터 유럽의 3개 대학과 10개의 유럽회사가 3년간 공동으로 수행한 과제이며 약 1500만 불의 비용이 소요되었다.

제어는 다음의 5 개 영역으로 구분될 수 있는 구성 객체(architecture object, 앞으로 AO로 명명함)의 집합으로 구성된다.

- **인간-기계간 제어 :** 외부 실체(예, 조작자)에 대한 기계 혹은 기계의 한 부분을 나타내며 이러한 실체들에게 기계 동작을 제어하게 한다.
- **동작 제어 :** 주어진 자유도의 상대적 동작을 기계가 만들어 내도록 한다,
- **논리 제어 :** 기계 내에 장착된 센서들에서 취한 데이터와 액츄에이터들의 동작을 담당한다.
- **축 제어 :** 축들이 동작 제어 모듈에 의해 만들어진 명령을 수행하게 한다.
- **공정 제어 :** 기계의 보조 시스템의 데이터 관리와 처리를 담당한다.

이러한 AO 각각에 포함되어지는 기능들의 양을 생각하면, 각 AO들은 더 나누어 질 수 있다. 즉 모듈의 재사용성을 높이며, 고 수준의 상호교환성을 얻기 위한 방법이다.

구조는 그림 1에 있으며, 플랫폼은 하드웨어, 운영체제, 통신 시스템, 선택부분인 데이터베이스와 그래픽시스템을 포함하고 있다. 플랫폼에 대한 서비스는 표준화된 API를 사용하여 이용되어 진다. 이 과제의 중요 부분이 제어 내부간 통신 부분이다. 이 부분을 잘 정의함으로써 벤더 중립형 통신 시스템을 규정하였다.

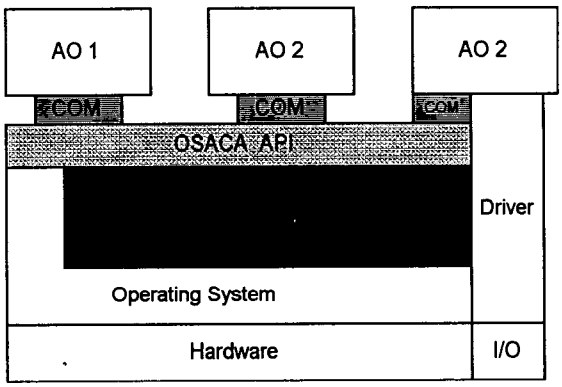


그림 3. OSACA의 구조

API(Application Programming Interface)를 살펴보면 C++ 언어를 사용하여 규정함으로써 모듈을 만들어 재구성할 때 약간의 제약성을 가지고 있다. 또한 새로운 머신 툴을 구축하는데 유연성과 사용자 친숙도를 높이기 위해 시스템 부팅동안 그리고 동작 중에 재구성을 가능하게 하였다. 이러한 과정을 처리하는 모듈, 즉 구성(컨피규레이션) 시스템이 필요한데, 이것은 OSACA의 독자 모델이다.

4.2 OMAC-TEAM(Open Modular Architecture Controller-Tech. Enabling Agile Manufacturing) [3-6]

1983년 미공군에서 발주한 Next Generation Controller (2000만불) 과제로부터 시작하여, Enhanced Machine Controller(100만불)과제를 거쳐 1994년에 미국의 개방형 구

조 제어기의 시금석이 되는 OMAC에 대한 요구사양이 제시되었다. 이를 기반으로 미국 DOE와 GM, Ford, Chrysler가 공동으로 개발하는 과제가 OMAC-TEAM이다.

제어 시스템은 핵심 모듈이라 불리는 다음의 11개의 기능적 모듈로 나누어진다.

- **작업 조정기 :** 동작들을 순서화하고 다양한 동작을 조정/감지하며 사건구동형 제어공정을 수행하는 모듈
- **파트(part) 프로그램 번역기 :** 파트 프로그램을 제어 시퀀스로 번역하는 모듈
- **축 그룹 :** 개별 축의 동작을 조정하는 모듈
- **축 :** 동작 제어를 하는 모듈
- **키네마틱스모델 :** 키네마틱스 구성과 계산용 모듈
- **제어 법칙 :** 동작 루프를 계산하는 모듈
- **인간 기계간 인터페이스 :** 데이터, 명령 및 내부 제어기 모듈의 이벤트를 원거리에서 처리하게 하는 모듈
- **공정 모델 :** 제어 시스템에 통합되어 지는 동적 데이터 모델을 포함하는 모듈.
- **이산 논리 :** 이산 논리 계산을 하는 모듈. 다중 이산 논리 모듈은 네트워크로 연결된 PLC들과 유사하다.
- **입출력 점 :** 일반적인 읽기/쓰기 인터페이스를 사용하여 입출력기기를 제어한다.
- **기계 대 기계(통신) :** LAN 및 필드버스를 통한 디지털 통신을 위한 모듈

구조는 그림 2에 있으며 인프라구조는 플랫폼서비스, 운영체제, 프로그래밍 도구를 포함한 환경 일체를 말한다. 여기서 플랫폼 서비스는 타이머, 인터럽트 핸들러, 프로세스간 통신 등을 처리한다. 인프라구조로는 DCOM, CORBA, OSACA 모델을 사용할 수 있지만 실시간성과 관련된 문제를 고려해야 한다.

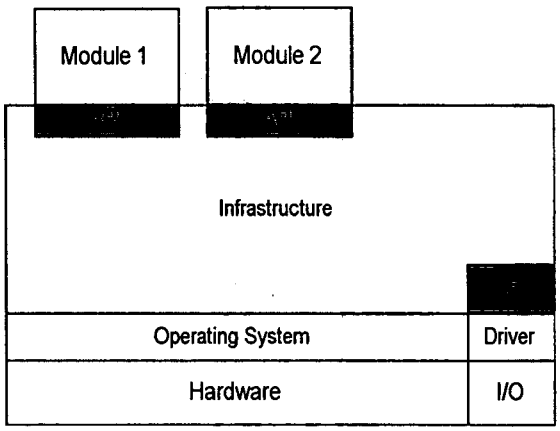


그림 2. OMAC의 구조

API는 특정 프로그래밍 언어와 무관한 IDL(Interface Definition Language)로 제공되면, IDL 컴파일러가 C++/C 및 자바로 된 API 프로그램을 만들어주기 때문에 제 3 자가 그들 자신의 모듈을 쉽게 개발할 수 있다. 이 과제는 이러한 모듈들을 재구성할 수 있도록 해주는 구성 도구에 대한 내용이 아직 규정되어 있지 않다.

4.3 OSEC(Open System Environment for Controller)[7,8]

1995년 12월에 일본에서 시작한 과제로 3개의 NC회사와 2개의 정보시스템회사와 하나의 제어기 빌더 회사가 모여 기반을 만들어 25개의회사와 15개 대학으로 확장하여 시작하였다.

제어 시스템은 5 개의 제어 모듈에 대응되는 5개의 기능군으로 나눌 수 있다.

- **인간 기계 인터페이스 :** 동작, 경보와 툴 관리를 담당
- **자원관리 기능 군 :** 자원관리 도구들은 동작 패널, NC 프로그램과 같은 데이터베이스, NC 언어 처리기 등일 수 있다. 이러한 자원들로부터 나오는 이벤트들을 다루고 머신 툴 모드에 따른 머신 툴을 관리한다.
- **동작 생성 기능 군 :** 파트 프로그램을 해석하는 동작을 생성한다.
- **기계 제어 기능 군 :** 상호보간, 가속 및 감속 등을 수행하고 서보 및 입출력을 동기화한다.
- **기기 제어 기능 군 :** 서보 모터 및 PLC단자로부터/에 입력/출력을 제어한다.

구조는 그림 3에 있으며, 기능 블록 기반으로 만들어져 있기 때문에 다양한 모듈을 만드는데 적합하다. 이 구조는 기능 블록에 따라 S/W와 H/W의 한 쌍으로 모듈 개념을 사용하는 제약점이 있다.

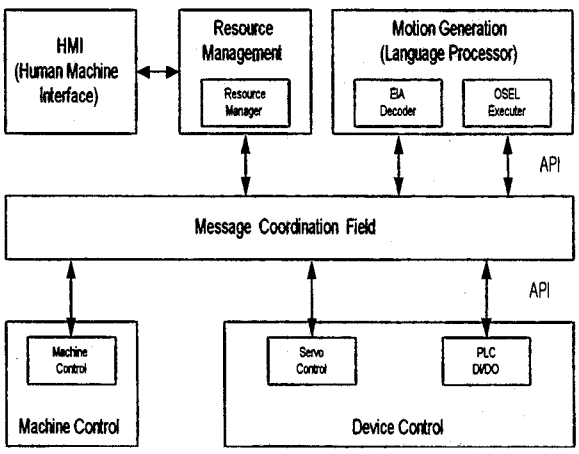


그림 3. OSEC의 구조

API는 기능성과 실시간 주기로 표현하는 제어기 소프트웨어 모듈간에 교환되는 메시지의 형태로 각 모듈에 대해 정해진다.

4.4 OPC(OLE for Process Control) [9]

1996년 가을에 140개 회사를 구성된 OPC foundation을 만듦으로써 시작하였다. 이 것은 앞의 과제들과는 달리 NC와 로봇제어기를 위한 개방형 제어기가 아니며 화학공정과 같은 플랜트에 사용되는 DCS를 위한 개방형 제어기를 표방하고 있다. 그러나, OPC의 개발 방향에는 PLC, NC, 로봇 제어기를 포함하도록 되어 있다.

OPC는 여러 다양한 공정 제어 관련 기기들 간에 데이터 통신을 위한 표준 인터페이스를 정의해 놓은 새로운 산업 표준이다. OPC는 Microsoft의 OLE/COM 표준에 기초를 두고 있으며, OLE/COM은 Microsoft에 의해 설계되어 졌지만 다른 사람들에 의해서 확장되어질 수 있다. 즉, OPC는 DCS, PLC, historian, 또는 소프트웨어간의 실시간 정보를 교환할 수 있도록, OLE기술 기반 위에 객체, 메쏘드, 성질들을 정의하고 있다.

OPC의 서버 구조는 그림 4에 있으며, 서버내의 응용 모듈은 COM 기반으로 설계될 수 있다. API의 호출은 클라이언트 컴파일 시에 미리 정의된 IDL(Interface Definition Language)에 의해 기술된 인터페이스를 참조하여 연결된다. OPC는 2 형태의 API를 정의하고 있다 하나는 고기능의 많은 데이터 전송을 요하는 어플리케이션을 대상으로 설계되었으며, OLE 특정인터페이스이고, C++로 개발되어지는 경우에 사용되어진다. 다른 하나는 OLE 자동 인터페이스로 데이터의 접근을 쉽게 하는데 주안점을 두어 만들어 졌으며, 이 인터페이스는 OLE 개념을 사용한 언어(Visual Basic) 및 응용 프로그램(Excel, Word의 매크로 기능)을 이용하여 데이터를 사용할 수 있다.

제어 기기 제조 업체의 경우는 새로운 기능 등의 추가나 삭제가 있을 때 즉, 기기의 사양에 변동이 있을 때마다, 클라이언트 제조 업체들은 드라이버를 수정해야 했으나, OPC는 클라이언트 쪽의 소프트웨어의 수정이 필요 없도록 서버와 클라이언트간의 인터페이스를 격리해 놓게 된다. 소프트웨어 응용 사용자 입장에서 보면, 기존에는 각 제어 기기마다 별도로 제공하는 인터페이스를 사용하여 소프트웨어를 만들어 왔으나, OPC를 사용함으로써 공통된 인터페이스를 이용할 수 있게 된다는 장점이 있다. 또한 OPC의 플러그 앤 플러그 기능을 이용하게 되면 시스템 통합에 더욱 많은 이득을 볼 수 있게 된다.

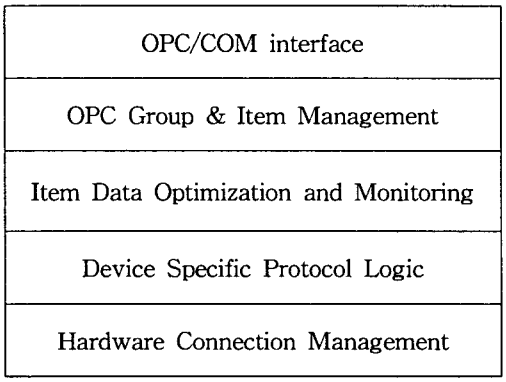


그림 4. OPC의 서버 구조

5. 전망 및 토의

본 고에서 최근에 수행했던 개방형 구조 제어기관련 과제들에 대해 살펴보았다. 본 고에서는 지면 때문에 많은 개방형구조 제어기에 대한 내용을 언급하지 못했다. 그러나, 대부분의 과제들은 개방형구조 제어기들에 대한 개념들 비슷한 반면 구현하는 데는 많은 차이가 있다는 것을 알 수 있다. 개념 등에 있어서 비슷한 것은 다음과 같다.

- *제어기의 기능적 분할을 통한 모듈형 참조 구조*
- *객체 지향 방법을 이용한 모듈 소프트웨어와 하드웨어 인터페이스*
- *모듈간 상호운용성과 통합을 가능하도록 플랫폼과 관계없는 통신 구조의 정의*
- *제어기를 동적으로 재구성하게 하는 구성 툴*

같은 개념을 가지고 객체지향방법을 사용하여 개발했지만 구현된 결과는 많은 부분에서 다르다. 즉, 정보 통신 기술(예, WEB, DCOM, CORBA 등)과 필드버스 기술들을 어떤 방향에서 적용하였느냐의 차이와 어떤 제어기를 대상으로 했느냐의 차이 때문이라고 본다. 결과를 가지고 보면, 가장 높은 수준의 개방성을 가진 제어기는 OPC라고 생각되지만 이 수준은 3 정도로 볼 수 있다. 그러나, OPC에서 사용한 방법을 NC, 로봇 제어기에 적용하는 것은 많은 연구가 필요하다. 이는 앞에서 언급한 실시간성 등과 같은 문제들 때문이다. OMAC와 OPC에서는 요즈음 사실상의 표준으로 진행되는 OLE(Object Linking & Embedding)에 기반을 둔 COM 개념을 사용하여 모듈을 개발하였고, OSACA와 OSEC은 자체 기준하의 객체를 정의하여 사용하였다. 따라서, 제 3자가 많이 참여할 수 있게 하는 방법은 전자의 방법이라고 생각된다.

다음의 4개의 제어기, CNC, PLC, 로봇 제어기 및 DCS등을 살펴보자. 이들 제어기의 H/W 관점에서는 모두 CPU 보드와 각종 입출력 보드를 가지고 있고 각 제어기의 특성에 따라 입출력의 종류 및 수가 다르다는 것을 알고 있다. 또한 소프트웨어 입장에서 살펴보면 대부분의 제어기가 PLC의 주 기능인 논리 제어 기능을 보유하고 있으며 각 제어기의 특성에 맞는 기능들을 사용하고 있다. 예를 들어 MMI 기능은 모든 제어기에 공통적인 내용이며, 작업 조정기(task Coordinator)는 CNC 및 로봇 제어기에도 사용한다. PID 제어 기능, 적응 제어 기능 등은 CNC, 로봇제어기에 사용되며 동시에 PLC 및 DCS에도 적용되고 있다. 이러한

공통적인 기능 때문에 OSACA, OMAC-TEAM, OSEC, OPC에서는 PLC 기능을 내포하도록 IEC 61131-3[12]을 지원하고 있다. 또한, OPC를 제외한 3과제는 모두 CNC와 로봇을 대상으로 개발하고 있다. 이러한 점 때문에 OPC는 CNC 및 로봇제어기까지 확장할 계획을 가지고 있다.

이외에 각 개방형 구조 제어기가 가지는 특이 내용은 OSACA, OMAC-TEAM, 및 OPC는 센서 및 액츄에이터용으로 필드버스 등을 사용하였고, OSEC은 이더넷 및 WEB을 이용한 사이버 공장의 개념을 사용하였다.

이러한 과제에서 빠져 있는 것은 모듈의 이식성에 대한 내용이 빠져 있다. 실제로 개방형 구조 제어기의 정의를 살펴보면 이식성은 상당히 중요하지만, 이를 강조할 때는 현재 기술로 보면 실시간성을 보장하지 못하기 때문이 아니냐라고 생각된다. 다른 종류의 CPU간에 모듈의 이식성을 보장하기 위해서는 자바와 같은 VM(Virtual Machine)을 사용해야 하기 때문이다.

요약하면, 앞으로의 개방형 제어기는 필드버스 기반 하에서 세 번째 수준의 개방형구조 제어기를 만드는 방향이 될 것이다. 이러한 개방형 구조 제어기는 PLC, DCS, NC, 로봇 제어기를 모두 포함하며, 모듈의 구성에 따라 모든 기능을 제공하거나 일부 기능을 제공하는 신축성 있어야 한다. 따라서, 세 번째 수준의 제어기는 2절에서 개략적으로 언급했지만, 실시간성을 요하는 경우 H/W등을 이용하지만 표준 구조를 따라가며, 대부분은 소프트웨어적으로 COM, CORBA 등을 이용함으로써 사용자는 IDL등을 통하여 쉽게 모듈 등을 만들고 재구성가능하게 만들어야 한다. 물론 시스템이 재구성되어 질 때 실시간성이 보장될 수 있도록 해야 하기 때문에 실시간성을 검사할 수 있는 툴들의 개발이 필요하다. 모듈의 이식성과 실시간성이 보장이 되는 VM이 개발될 것으로 보인다.

참고 문헌

[1] Fisher-Rosemount, PlantWeb Performance Seminar, Mar. 1999.

[2] OSACA, http://www.osaca.org/

[3] OMAC, http://www.arcweb.com/omac/

[4] OMAC-TEAM, http://isd.cme.nist.gov/info/teamapi/

[5] Chrisler, Ford Motors. GM, "Requirements of Open, Modular, Architecture Controllers for Applications in the Automotive Industry V.1.1, 1994.

[6] J.L. Michaloski, S. Birla, R.E. Igou, H. Egdorf, C. J. Yen, D.J. Sweeney, G. Weinert, " The TEAM API open Architecture Methodology," 1998.5

[7] OSEC. http://www.sml.co.jp/OSEC

[8] S. Fujita, M. Kanemoto, H. Asano, T. Inoue, A. Okano, H. Uno, N. Kakishita,"OSEC: Open System Environment for Controllers. 1998.5

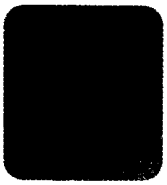
[9] OPC, http://www.opcfoundation.org/index.html

[10] F. M. Proctor, "Practical Open Architecture Controllers for Manufacturing Applications," 1998.

[11] Y. Koren, "Open-Architecture Controllers for Manufacturing Systems," 1998.5

[12] IEC, Programmable Controllers - Part 3 Programming Language, IEC61131-3, 1993.

저 자 소 개



박홍성(朴洪聖)

1961년 3월 16일생. 1983년 서울대 공대 제어계측공학과 졸업. 1986년 서울대 대학원 제어계측공학과 졸업(석사). 1992년 동 대학원 제어계측공학과 졸업(공박). 1996년-1997년 독일 아헨공대 박사후 연구원. 현재 강원대 공대 전기전자공학부 부교수