기술정보

분산제어 시스템의 이론과 실제(Ⅲ)

글/최 병 욱(금성산전 연구소 선임연구원)

<전월호에서 계속>

4.6 MAP(Manufacturing Automation Protocol)

고급 품질의 유지와 생산송 향상은 컴퓨터와 다른 자동화 장비들에 의하여 구현된다. 그러나 이러한 구현이 생산성을 향상할 수 있다고 하여도 고가이고 시장성이 있으며 최적의 자동화 시스템을 구현하는 데는 여러가지 문제점이 존재한다. 이는 공정의 특성에 맞추어서 여러 회사의 제품을 조합하여 사용하여야 한다는 문제가 발생한다. 따라서 접속에 많은 비용과 부가장비가 필요하게 된다.

이러한 문제를 해결하기 위하여서는 자동화 시스템에 있어서 제품에 무관한 통신접속방법이 요구된다. 1980년에 GM에서 처음으로 여러회사의 자동화 장비를 접속할 수 있는 통신방법에 대한 연구인 MAP Task Force가 시작되었다. 이 당시에 GM에는 약 20,000PLC와 2,000robot가 설치되 있었는데 통신 및 접속에 필요한 배선 및 케이블 접속의 절감으로 약 50% 의 설치비가 감소되리라 예상되었다.

이와 같이 자동화 설비의 통신 및 접속의 표준화는 많은 요구가 분출되었으며 1984, 1985년에 사양이 발표되었다. 실제로 MAP사용자 그룹에는 대부분의 기업이 동참하였으며 실제로 1980년은 MAP 개발의 시작이라 하겠다. 이시기는 앞서 설명하였듯이 ISO의 OSI 기준모델의 표준화가 확실시 되었고 IEEE 802 프로젝트도 LAN에서의 표준화를 제정하던 시기였다. 따라서 MAP의 필요성이 강조되기 시작하였다.

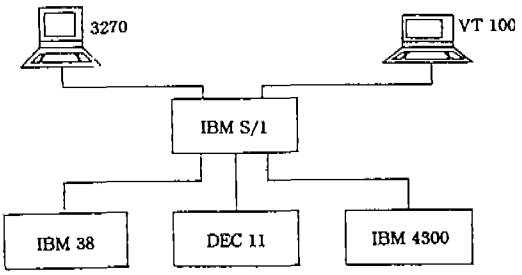
MAP 2.1 버젼에서는 다음과 표준화안이 적절한 방법으로 채택되었다.

7. ppl;ication layer : FTAM(File Transfer and Access Management), 즉 ISO DP 8571, GM MMFS(Manufacturing Messaging Format Standard)
6. Presentation layer : Nil(ASCⅡ 및 Binary Encoding)
5. Session layer : ISO IS 8327
4. Transport layer : ISO DIS 8073
3. Netmork layer : ISO DIS 8473
2. Data link layer : DIS 8802/2(ISO Logic Control) 즉 IEEE 802.3
1. Physical layerJISO DIS 8802/4 (ISO Token Passing Ring) 즉 IEEE 802.4(Token passing bus Media Access Control)

전송 선로로는 boradband 동축케이블이 권유되었으며, 10Mbps의 전송속도가 구현되었다. 이와 같은 MAP의 구현을 위하여 다음과 같은 실현단계가 계획되었는데 어떠한 것은 현재 구현된 상태에 있다.

1) 단계 1

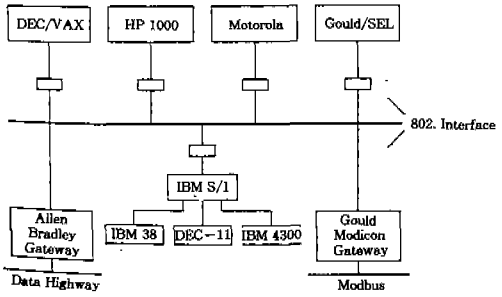
중앙집중식 네트워크로 1984년에 완성되었다. 그림 4.15와 같이 IBM S/1에 여러 가지의 컴퓨터(IBM 38, IBM 4300, PDP 11, HP 1000)가 접속되었다.



<그림 4.15> MAP－단계 1

2) 단계 2

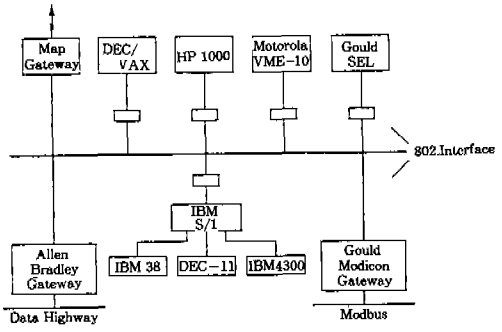
Local area network로 1984년에 그림 4.16이 완성되었는데 3단계로 실현되었다.



<그림 4.16> MAP－단계 2

3) 단계 3

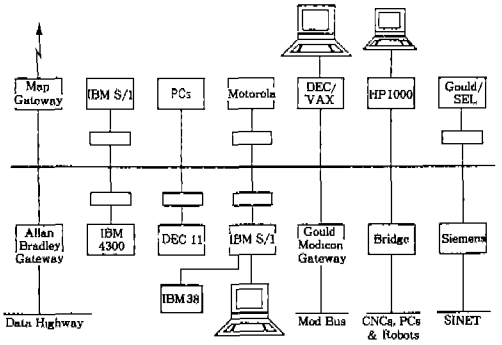
응용서비스로 1985년에 완성되었는데 여기에서 gateway가 개발되었으며, 단계 2의 LAN이 발전하였고 소프트웨어가 상당히 보강되었다.



<그림 4.17> MAP－단계 3

4) 단계 4

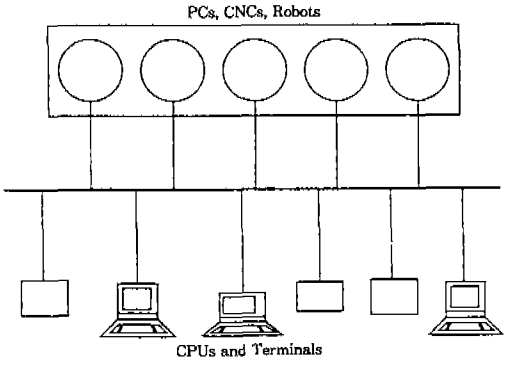
개발된 소프트웨어를 LSI화 및 마이크로 프로세서화하여 낮은 가격의 하드웨어를 개발하였다. 소프트웨어가 적절한 하드웨어로 구현되었고 ISO 레이어 1－4가 하드웨어로 구현되고 레이어 5가 완성되었으며 레이어 6이 첨가되었다.



<그림 4.18> MAP－단계 4

5) 단계 5

완성된 네트워크 유틸리티의 개발로 1986으로 계획되었다. 이 단계의 MAP 실현의 최종단계로 이로써 여러가지 제품이 통일된 형태로 접속이 가능하게 되었다.



<그림 4.19> MAP－단계 5

5. 소프트웨어

기본적으로 자동제어 컴퓨터 시스템에서 소프트웨어는 시스템 소프트웨어와 응용 소프트웨어로 구분된다. 분산제어 시스템에서 이 두가지의 분류는 통

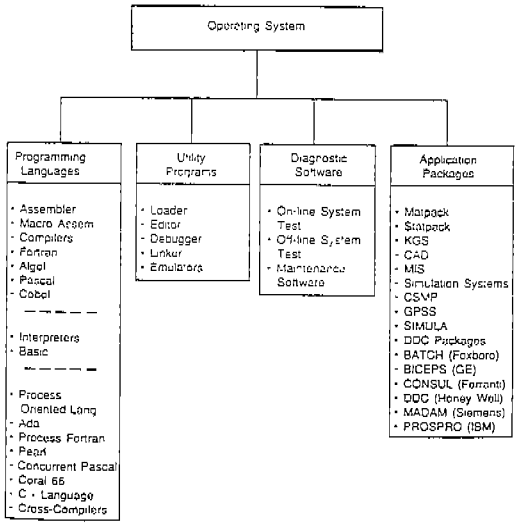
상 혼재한다. 그리고 통신 소프트웨어와 구조 및 매개 변수 소프트웨어와 같은 분류도 동일하게 된다. 더구나 제어와 진단 기능의 소프트웨어가 공존되기 때문에 전문가 소프트웨어가 하나의 중요한 요소가 된다.

일반적으로 컴퓨터에서 사용가능한 소프트웨어를 시스템 소프트웨어라 하며 결과적으로 말하면 새로운 시스템의 개발을 위한 도구일 뿐아니라 응용 프로그램의 사용을 위한 준비과정을 지원하는 소프트웨어로서 컴퓨터 회사가 제공하는 것이다. 분산제어 컴퓨터 시스템에서 필요한 시스템 소프트웨어는 그림 5.1과 같으며 크게 보아서 다음과 같은 특성을 보유하여야 한다.

1) 실시간 운영 시스템

2) 프로세스 지향적인 프로그램 언어

3) 유틸리티 프로그램 및 프로그램을 할 수 있는 도구



<그림 5.1> 시스템 소프트웨어의 구성

5.1 실시간 운영 시스템

운영 시스템은 시스템 소프트웨어로서 컴퓨터 시스템 예를 들면 프로세서, 메모리 및 입출력 소자등의 자원을 관리하고 운영하는 시스템 소프트웨어이다. 이것은 성능을 유용하게 관리하며 최적화하고 또한 사용자가 쉽게 응용 프로그램을 개발할 수 있도록 지원한다.

실시간 운영 시스템은 실시간 자원을 관리하고 제어하는 운영 시스템으로 고속을 요하는 자동제어 시스템에서 실시간으로 자원의 처리 및 제어를 위하여 필요한 시스템 소프트웨어이다. 따라서 실시간 운영 시스템을 사용할 경우는 입출력 소자에 대한 임계치에 적절히 대처할 수 있는 시스템 제어 특성을 보유하여야 한다.

따라서 실시간 제어용 운영 시스템은 다음과 같은 특성을 고려하여 개발되어야 한다.

1) 프로세서 시간 관리－타스크 스케줄링·타스크 생성 및 처리, 동기화 및 종결

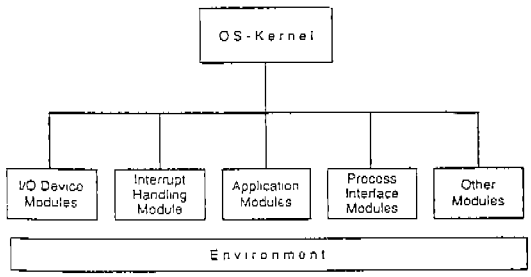
2) 메모리 관리－다중 프로세스 시스템에서 프로그램의 재할당(Realocation) 가상 어드레싱(Virtual addressing) 및 페이지 및 세그먼트 등의 관리

3) 주변 소자 관리－입출력 소자, 화면 및 콘솔 그리고 하드 디스크의 처리 등 주변 소자의 제어 및 스케줄링

4) 데이터 베이스 관리－통상적인 화일 관리뿐아니라 대용량의 데이터 베이스 관리

5) 프로세스 인터럽트 처리－프로세스 제어 장비로 부터의 인터럽트 신호의 응답 및 처리

그림 5.2는 실시간 운영 시스템의 일반적인 구조로서 가운데 부분을 커널이라고 하며 외부 및 내부의 인터럽트 처리, CPU 시간의 디스팻치, 우선순위에 기초한 타스크의 제어 및 동기화 처리 등을 수행한다.

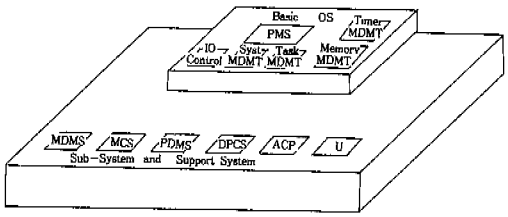


<그림 5.2> 실시간 운영 시스템의 일반적인 구조

한편 분산제어 시스템으로의 응용을 위하여 실시한 제어용 운영 시스템이 요구된 타스크의 응답이나 수신된 인터럽트의 처리 등의 사용자 목적에 맞는 시스템이 개발되었는데 그림 5.3은 하나의 예를 나

타낸다. 그림에서 가장 큰 두가지 특징은 다음과 같다.

1) 기본적인 운영 시스템
   메모리 관리
   시스템 관리
   타이머 관리
   타스크 관리
   I/O 제어
2) 부시스템 및 지원 시스템
   분산 프로세스 제어 시스템(DPCS)
   자동제어 프로세스(ACP)
   프로세스 데이터 관리 시스템(PDMS)
   사용자 편이 통신 시스템(MCS)
   사용자를 위한 데티타 관리 시스템(MIDMS)
   수학 라이브러리(AL)



<그림 5.3> HIAC-300의 운영 시스템

5.2 통신 소프트웨어

통신 소프트웨어는 각각의 시스템간이나 데이터 베이스간의 데이터 정보에 사용되며 버스나 LAN을 이용하게 된다. 최근의 추세는 앞에서 설명한 ISO/OSI 모델을 사용하나 레이어를 모두 사용하지는 않는다. 유연한 다중 레이어 구조를 유지하는 소프트웨어이기 위해서는 레이어간 서로 독립적인 구조를 이루어야 한다. 이런 구조만이 자유로운 구조 변경이나 대치가 가능하게 된다.

MOD300 시스템을 예를 들어 설명하면 다음과 같은 레이어가 사용된다.

1) 응용 레이어(Application layer)
사용자의 응용 프로그램과의 접속

2) 전송 레이어(Transport layer)
데이터의 전송, 에러 처리 및 복구 그리고 메시지 전송의 확인

3) 네트워크 레이어(Network layer)
서로 다른 통신 소자간의 메시지의 물리적인 변환

4) 데이터 링크 레이어(Data link layer)
송신단에서 수신단으로 전송될 데이터를 위한 데이터 포맷의 형성

5) 물리적 레이어(Physical layer)
물리적인 전송 선로를 통한 메이시의 전송을 위한 기능적인 면이나 물리적인 면의 구성

5.3 프로그래밍 언어

50년대 초의 어셈블리 언어의 사용에서 시작된 프로그래밍 언어는 현재 프로세스 중심의 언어로 발전되어 왔다. 이 기간 동안 다양한 언어의 개념과 함께 응용 분야 및 기술이 상당한 발전을 거듭해 왔다. 그림 5.4와 같은 언어의 발전을 나타낸다.

현재 응용 예제에서 요구되고 있는 고차원 언어의 특성을 보면 다음과 같다.

1) 발생된 코드의 최적화
2) 복잡한 데이터 구조
3) 다이나믹 메모리 저장 기법
4) 자체 도큐멘테이션 및 향상된 유틸리티 프로그램에 의한 지원
5) 수정이 용이한 구조 및 신뢰성 있는 프로그램
6) 번역기의 특성을 포함한 구조(Interactive features)

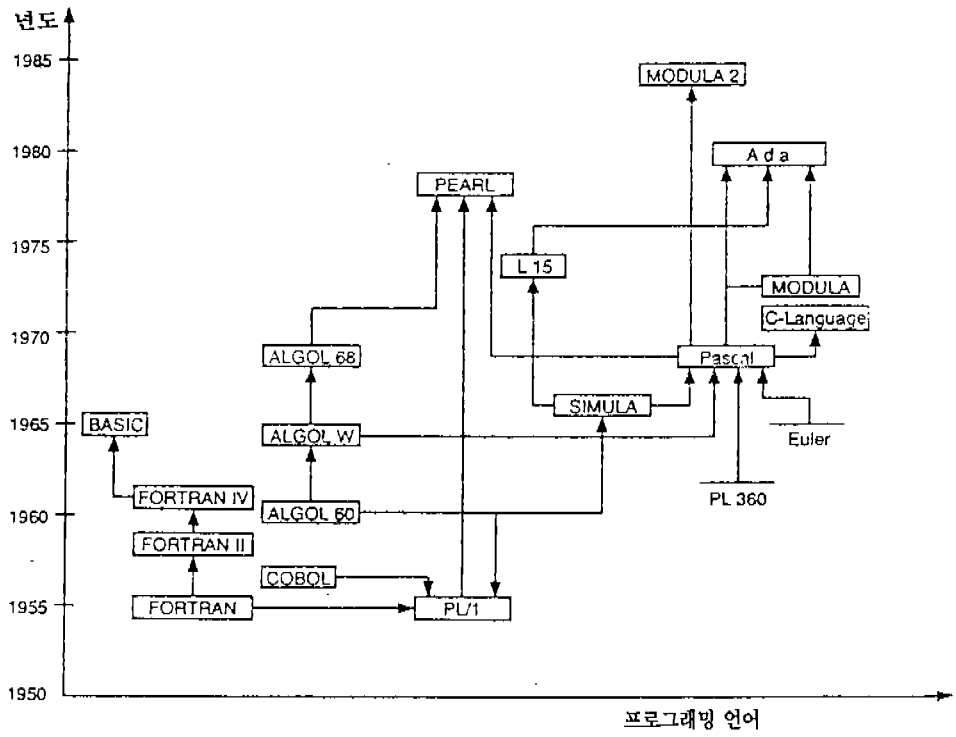
5.3.1 고기능 프로그래밍 언어의 특성

현재 사용화된 프로그래밍 언어는 매우 다양하며 따라서 사용자가 오히려 언어의 선정에 혼란이 오는 현상이 있는 정도이다. 문제의 해결은 초기에 컴퓨터 설계자에 의해 최적화된 구조로 시작된다. 따라서 분산제어 시스템에서의 소프트웨어는 응용 프로그램의 개발자나 컴퓨터 설계자간의 유연한 협력에 의하여 일반적인 구조로 형성되어야 하며 최적화 되어야 한다.

여기서는 여러 프로그래밍 언어 중에서 자동 제어용으로 많이 사용되는 가지에 대하여 간단히 특성을 살펴보도록 하겠다.

1) FORTRAN

FORTRAN 언어는 과학적 계산을 위하여 가장 많이 사용되어 온 언어이다. 1950년대 중반에 개발된 최초의 High-level language였으며 1966년에 표준화가 되었다. 이후 여러번의 개정을 거쳐서 1970년대에 최신 구조인 Fortran 77이 완성되었다.



<그림 5.4> 프로그래밍 언어의 발전

이 언어는 메인 프로그램과 서브루틴의 조합으로 구성된다. 그리고 수행을 위하여 컴파일 과정과 링크 과정이 필요하게 된다. 그러나 단지 수의 표현(Integer, Real, Complex, Double Precision Real)과 이진 데이터, 어레이 그리고 문자 및 파일이 허용되었으며 두가지 레벨(Global, Local)의 변수 구성이 가능하다. 초기에는 단지 Batch-Mode의 조작만이 가능하였으나 최근에 실시간의 메모리 관리가 가능한 실시간 FORTRAN이 개발되었다.

2) BASIC

BASIC 역시 FORTRAN과 마찬가지로 문제 중심의 언어로서 과학적 계산이나 엔지니어의 응용 프로그램에 많이 사용되어 왔다. 또한 문제의 해결을 위한 언어의 구조는 FORTRAN과 같이 공식과 유사한 모습이 된다. 그러나 FORTRAN은 컴파일러 언어이지만 Basic은 인터프리터 언어이다. 1980년 초기에 공장 제어를 위한 실시간 BASIC이 개발되었다.

3) PASCAL

PASCAL 언어는 블럭 구조의 언어로서 교육 및 컴파일러 설계를 위하여 설계되었다. 따라서 PASCAL 컴파일러 자체가 PASCAL로 구현되었다. 구조를 보면 메인 블럭과 서브루틴 블럭으로 구성되며 다음과 같은 일반적인 구조를 갖는다.

- Header
- Definitions of Constans
- Definition of Types
- Declaration of Local Variables
- Subroutine Definitions
- Execution Statements

사용 가능한 데이터 종류는 Integers, Real, Characters, Enumeration, Booleans, Arrays, Records, Sequencial file, A Limited Form of Set이며 내부 스택을 중심으로 수행된다.

3) C

C언어는 UNIX 운영 체계를 개발하기 위하여 D. M.Ritchie와 B.W.Kernighan에 의하여 개발되었다. 따라서 시스템 소프트웨어의 개발을 위한 언어로 사

용가능하며 또한 응용 프로그램의 개발도 가능하다. 예를 들면 입출력이 직접적인 함수의 사용이 아니라 관계된 서브루틴의 사용으로 다음과 같은 형태로 간단히 해결된다.

```
main()
{
        Printf("C language\n");
}
```

실시간 데이터 처리의 측면에서 보면 C언어는 매우 다양한 제어 형태를 제공하여 서브루틴, 보조함수, 병렬 처리 기법의 처리 및 동기화 등을 지원한다. 더우기 C언어 컴파일러는 비교적 간단하고 요약되어 있어서 어셈블된 C언어가 매우 효율적이며 소스프로그램과의 혼합이 가능하고 신뢰성이 양호하다. 따라서 실시간 제어용 응용 프로그램으로 가장 많이 사용되고 있다.

C언어는 구조적 특성을 갖는 언어로서 하나의 함수는 여러 곳에서 사용 가능하며 처음에 시작하는 서브루틴은 항상 main()이어야 한다. 따라서 C언어의 수행은 운영체계가 제어를 main()에게 이양함으로써 실시된다. C언어의 구조화된 특성은 다음의 예와 같다.

```
main()
{
}
Function A()
{
}
Function B()
{
}
Function C()
{
}
```

그리고 입출력 선언 등은 공통의 자료화로 관리가 가능한데 Include 구조에 의한 Header File의 형태로 여러 화일에 공유될 수가 있다. 또한 상수와 같은 선언 Define 문장에 의하여 추상화가 가능하다.

4) PL/M

PL/M 언어는 INTEL 마이크로 프로세서를 위한 언어로서 설계되었다. 이 언어에서 생성된 프로그램은 운영 체계를 필요로 하지 않는다. 따라서 하드웨어 중심이며 블럭 및 구조화된 제어 구조를 지원하기 때문에 결과적으로 나타난 언어가 매우 투명하며 모듈화가 가능하다. 데이터 형태로는 BYTE, WORD, DWORD, INTEGE, REAL, POINTER 및 LABEL이 지원된다.

5.3.2 프로세스 지향 프로그래밍 언어의 특성

프로세스 지향 프로그래밍 언어는 콘커런트(Concurrent)수행 능력 및 실시간 프로그램이라고 하는 다중 타스크의 수행 능력을 특징으로 하고 있다. 가장 널리 알려진 두가지는 Ada, Pearl이다.

Ada는 여러명의 프로그래머에 의하여 작성되는 매우 큰 소프트웨어를 개발하는 것을 지원하기 위하여 개발되었다. 데이터 형식은 Pascal 등에서 사용된 모든 것을 지원하고 있으며 콘커런트 프로그램이 가능한 구조를 갖는다.

Pearl은 산업 공정 제어를 위한 프로그램에 적합한 구조로 되어 있어서 실시간 프로그램의 다중 타스크구조와 프로세스 입출력을 위한 확장된 기능을 특징으로 한다. Pearl에서의 프로그램은 시스템 파트와 제어 알고리즘이 구현된 Problem Part로 구성되어 있으며 FIXED, FLOAT, BIT, CHAR, CLOCK 및 DUR(ATION)의 데이터 형식이 지원된다.

5.4 응용 소프트웨어

표준적인 공정 관리 및 제어 프로그램 외에 DCCS 공급업자가 제공하는 응용 소프트웨어가 있는데 이는 좀 더 복잡한 문제에 사용된다. 여러가지의 응용 소프트웨어가 제공되는데 이는 공급업자에 따라 조금씩 다르며 일반적으로 다음과 같다.

1) 진단 및 처리(ASEA－Master, CENTUM, DCI 5000, LOGISTAT CP－80, MDC－200, MOD 300, NETWORK 90, P 4000, PROCONTROL I, SPECTRUM, TDC 3000, TELEPERM M, YEWPACK Ⅲ)

2) 순차적 제어(CONTRONIC P, DCI 5000, MOD 300, NETWORK 90, P 4000, RS－3, TDC 3000 TELEPEM M)

3) 최적 및 적응 제어(ASEA－Master, DCI 5000, LOGISTAT CP－80, MDC－200, NETWORK 90 PROCONTROL I, SPRCTRUM, TELEPERM M)

4) 고차원 알고리즘

5.5 소프트웨어 구조

5.5.1 문제의 정의

앞에서 언급한 유용한 소프트웨어는 응용 소프트웨어를 사용하기 위하여 완성된 형태로 제공되는 것이 아니라 응용 소프트웨어의 구조를 도와주는 하나의 방법론을 제공한다. 이러한 과정을 소프트웨어 구조의 결정이라고 하며 결정된 구조를 변경하는 것을 소프트웨어 구조 변경이라고 한다.

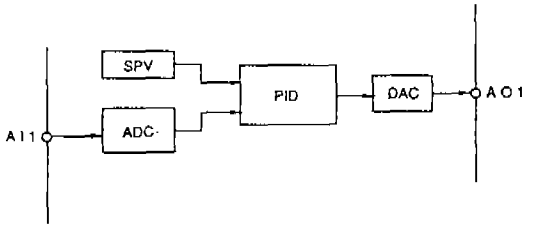
소프트웨어의 구조라고 하는 것은 소프트웨어 모듈을 이용하여 각각의 제어 루프를 연결하고 각각의 제어 문제에 적합한 제어 변수를 선정하는 것이다. 예를 들면 그림 5.5와 같이 간단한 제어 루프를 구성하기 위하여는 3개의 기능적인 부분을 서로 결합하여야 한다. 즉

ADC (Analog to Digital Conversion Block)

PID 제어부

DAC (Digital to Analog Conversion Block)

또한 제어 입력 AI1 및 출력 AO1이 연결되어야 하며 목적치는 컴퓨터나 다른 입력에 의하여 제공되어야 한다. 이와 같은 구성에서 보면 하드웨어 뿐아니라 소프트웨어가 구성되어야 한다.



<그림 5.5> PID 제어 루프의 구성

5.5.2 기능적 블럭 및 함수 라이브러리

복잡한 시스템에서 기능적으로 분리된 수 많은 블럭이 서로 연결되어야 하며 필요한 입출력이 연결되어야 한다. 소프트웨어 구성을 위하여 사용된 기능 블럭은 잘 알려진 제어 알고리즘을 구현하는 소프트웨어 모듈이 될 수 있다. 따라서 이러한 기능은 파괴되지 않는 메모리(ROM 또는 EPROM)에 저장되어서 DCCS의 라이브러리 함수로 사용된다.

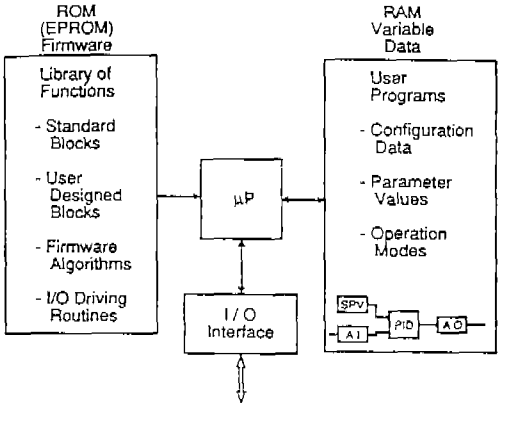
일단 구성된 소프트웨어는 향후 응용 목적에 비추어서 동작되기 위하여서는 제어 변수 예를 들면 PID 제어기의 (Kp, KI, KD) 게인 등을 선정하여야 하는데 이와 같은 작업을 시스템 매개 변수화라고 한다.

이와 같은 소프트웨어의 선정 및 매개 변수화는 제어 문제의 해결을 위하여 제어 엔지니어가 설정하여야 한다. 따라서 이러한 과정에서 어떠한 프로그래밍 언어의 지식도 불필요하게 된다.

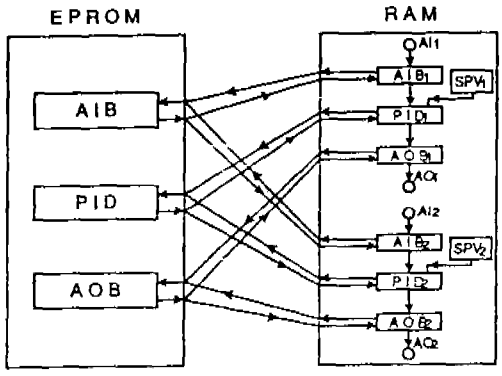
5.5.3 소프트웨어의 분리

응용 소프트웨어의 구성에 사용된 라이브러리 함수는 통상 ROM이나 EPROM에 저장되어 DCCS의 Firmware로서 사용된다. 그리고 매개 변수화나 구조화 과정에서 발생한 데이터는 RAM에 저장되어서 응용 소프트웨어의 수행에 사용된다.

그림 5.6에서 보면 소프트웨어의 구성이나 기능적 모듈이 모두 ROM에 저장되어서 모든 함수에 공용으로 쓸 수 있는 라이브러리 구조를 가지고 있음을 알 수 있다. 그러나 필요에 따라서 응용 소프트웨어의 구성을 변경할 필요가 있으며 예를 들면 포인터와 같은 구조에 의하여 소프트웨어의 수행이 변화할 수 있는 구조를 Softwiring이라고 한다. 이것은 각각의 기능적 블럭이 그림 5.7과 같이 재사용 가능(Re-entrant)구조임을 의미하며 구조 변경에 사용되는 변수는 변경이 가능한 구조여야 함을 의미한다. 그러나 이와 같은 데이터가 손실을 입는 경우를 대비하여 비상사태에 대응하는 데이터는 ROM에 저장되어야 한다. 따라서 시스템의 고장이 치유된 후에는 이와 같은 데이터가 메모리로 다운로딩되어 재실행이 가능하여야 한다.



<그림 5.6> DCCS에서 소프트웨어의 분리



<그림 5.7> DCCS에서 다중 사용되는 소프트웨어 모듈

5.6 지능 기법 소프트웨어

인공 지능(Artificial Intelligence)은 60년대 컴퓨터 공학에서 처음 제시된 이래 알고리즘에 의하여 해결되지 않는 문제나 알고리즘이 복잡한 문제와 같이 결정되지 않은 변수가 많은 문제를 해결하는 방법으로 널리 사용되고 있다. 공학도 입자에서 지능 제어의 문제는 감시제어. 고장진단, 적응 제어 및 자체 구조 변경(Self-Tuning)제어에서 사용이 가능하다.

현대의 분산제어 시스템에 있어서 지능 기법은 감시 제어 등의 문제 뿐아니라 최적 공정 제어 등 사용자의 요구가 급증하고 있다. 이러한 지능 기법은 예상할 수 없는 상황의 발생에 대한 처리나 예측을 가능하게 한다.

지능 기법의 소프트웨어는 일반적인 기법과 동시에 사용될 수 있으며 시스템의 지적능력을 향상시킬 수 있다. 최적 공정 제어 문제와 같이 생각해 보면 지능 기법은 공정의 최적 조건을 찾아 낼 수 있는데 이는 수학적으로 확인이 불가능한 경우 통계적 방법이나 Heurisric을 이용하는 경우 더욱 유용하게 된다. 또한 이러한 상황은 수치적으로 해석된 최적 조건 보다는 훨씬 더 빈번하게 발생된다. 그러나 최적의 공정 제어를 위하여서는 수학적 모델과 지적 기법이 적절하게 사용되어야 한다.

5.6.1 전문가 시스템

전문가 시스템은 숙련된 작업자의 조작과 같이 수학적인 모델링이 불가능한 지식 영역에서의 문제를 해결하는데 유용한 방법이 된다. 또한 숙련자와 같이 학습을 통하여 지식이 축적되어 더욱 효과적인 제어를 수행 할 수도 있다. 이러한 지식의 확장은 기존의 지식을 재구성하여야 하는 등의 변화를 요구하지는 않는다. 이는 숙련공이 새로운 지식의 습득이 기존 지식의 재구성을 요구하지는 않는 점과 동일하다.

사람과 비교하여 보면 전문가 시스템에서는 지식의 사실과 규칙으로 구성된 지식이 사실의 연결을 지정하는 것으로 볼 수 있다. 따라서 주어진 문제에서 전문가 시스템은 해결하고자 하는 문제에 필요한 사실과 지식을 사용하게 된다. 이러한 기법을 Pruning이라고 하며 관련없는 지식이 전체 문제를 해결하는 데 사용되어서 발생하는 손실을 제거하게 된다.

문제를 해결하기 위하여 사람이 사용하는 방법과 같이 전문가 시스템도 다음과 같은 두 가지 부분을 사용한다.

- 지식 기반(Knowlege Base)
- 추론 기관(Inference Engine)

결과적으로 전문가 시스템을 구현하는 데 있어서 해결하여야 할 두가지 문제는

- 지식을 어떠한 방법으로 표현하는가
- 어떠한 추론 방법이 사용되는가

에 있다.

1) 지식 표현의 방법

지식 표현의 방법은 지능 기법 소프트웨어의 일부분으로서 지정된 지식과 사실을 어떠한 방법으로 구조화하여 수학적인 계산이 없이 지식을 찾아내는가에 주된 목적이 있다. 인공 지능에서 어떠한 사실, 즉 지식은 문법적 관점과 구문적 관점에 의하여 표현된다. 문법적 구문은 지식이 어떠한 심볼에 의하여 어떤한 방법으로 표현되는가의 문제이며 구문적 관점에서는 지식이 어떠한 뜻으로 해석되느냐의 문제이다. 따라서 지능 기법을 사용하는 사람은 부호화(Encruption)보다는 지식의 표현에 좀 더 익숙하여야 한다.

현재 많이 사용되는 지식 표현의 방법은 다음과 같다.

- Production Rules
- structured Objects
- Predicate Logic

2) 추론 기관

전문가 시스템이 통상 사용되는 응용은 형상 인식

과 같이 가정과 인식에 의하여 가능한 해답에서 가장 최적인 해결책을 찾는 문제이다. 가정된 결론은 룰 해석기에 의하여 각 단계에서 효용성이 측정되며 Stimulus-Driven이나 Goal-Directed와 같이 전진형 방법과 후진형 방법에 지식간의 연결이 수행된다. 따라서 최종적으로 지식의 연결에 의하여 결론에 도달하게 된다.

이러한 전문가 시스템의 지식 표현 방법이나 추론 방법은 여기서 언급하기에는 내용이 복잡하므로 참고 문헌을 참조하기 바란다.

5.6.2 실시간 응용

전문가 시스템은 약 30여년의 역사가 있는데 스탠포드 대학에서 Heuristic Programming Project로 시작되어 현재는 여러가지의 기술적인 분석에서 보더라도 많은 성공을 이룩하였다. 현재는 여러가지 응용분야에 특히 하드웨어 구성이나 오프라인 데이터 분석 등에 사용되고 있으며 성공적인 예도 많이 존재한다. 그리고 시스템 진단이나 감시를 위한 온라인 시스템의 개발에 연구가 집중되고 있다. 실질적으로 공장 제어에서의 전문가 시스템의 응용에는 제한점이 있는데 문제의 복잡성에 제한이 있으며 공정의 진행이나 변화에 대한 시간적인 제약이 존재한다.

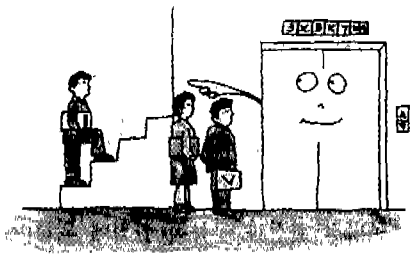
<다음호에 계속……>

에너지 절약 실천요령

엘리베이터의 격층 운행

●3층이하 운행금지

●4층이상 격층운행



시행효과

◆전국 승강기 설치대수 : 35,000대

◆총설치비용량 : 75만kW

◆사용시간 : 1시간/일×365일/1년

◆격층운행시절감율 : 10%

• 절 감 량 : 70만kW×1시간/일×365일/년×10%=
25,550천kWh/년

• 절감금액=25,550천kWh/년×81원/kW=20.7억원/년

공조기 설비에 변풍량(V.A.V)시스템채택

●공조설비에 변풍량(V.A.V)시스템 설치

시행효과

◆AHU수량 : 30,000

◆용 량 : 5kW

◆이행율 : 30%

• 절감전력량=$5\times30{,}000\times0.2\times540h=4.9\times10^6kWh$

• 절 감 금 액=$4.9\times10^6\times81$=4억원/년

냉수펌프의 효율적인 운전

●순환펌프에 변수량(V.W.V)시스템 채택

●노후 저효율 펌프의 교체

●과용량 펌프의 임펠러 가공 또는 회전수 제어

●가능한 자연수압 이용
(고가수조 펌핑의 경우 3~4층정도는 상수도 수압을 이용토록 배관개선)