□ 特輯 : μ-Processor應用

실시간 처리 컴퓨터 시스템

金 明 桓
(韓國科學技術院 敎授)

崔 殷 鎬
(韓國科學技術院 博士課程)

■ 차 례 ■

1 서 론

초기의 컴퓨터는 자루한 수치계산을 목적으로 개발되었으나, 반도체와 그 주변 기기의 발전으로 하드웨어의 가격이 저렴해지면서 여러 응용분야로 확산되었다.

실시간 처리를 요구하는 응용분야가 다른 컴퓨터 응용분야에 비하여 늦은 이유는 가격, 신뢰도, 접속장치 등에서 많은 문제점을 가졌기 때문이다.

최근에 하드웨어의 가격이 낮아지고 성능이 향상되면서 이같은 문제점들이 점차로 해결되어 공장의 자동화 등에서 실시간 처리를 요구하는 응용분야로 확산되고 있다.

실시간 처리에 주안점을 두는 시스템은 여러가지 특성들을 필요로 하므로 이에 적합한 컴퓨터를 사용해야 한다. 본지에서는 이같은 여러 특성과 이에 적합한 컴퓨터 구조 및 소프트웨어를 다루고, 실시간 처리를 위한 시스템의 앞으로의 전망을 고찰한다.

2 실시간 처리의 개요

2.1 실시간 처리의 정의

초기의 컴퓨터는 주로 batch 처리로 수행되다가 터미널 기기의 발전에 힘입어 on-line 처리가 널리 사용되었다.

On-line 처리는 크게 실시간 처리와 일반적인 데이타 처리로 나눌 수 있다. On-line 처리는 시스템 구조 면에서 그림 1과 같이 3부분으로 나눌 수 있다.

- ○ Sensor-based 처리 - A/D 또는 D/A 변환기와 같은 외부 접속장치를 통하여 물리적인 process 의 상태를 측정하고 필요한 제어를 한다.
- ○ Clock-based 처리 - clock을 통하여 특정 시간간격에 관계되는 처리를 수행한다.
- ○ Interactive 처리 - 터미널, 컴퓨터등의 외부장치와 연결되어 적합한 시간에 응답을 요구하는 일을 처리한다.

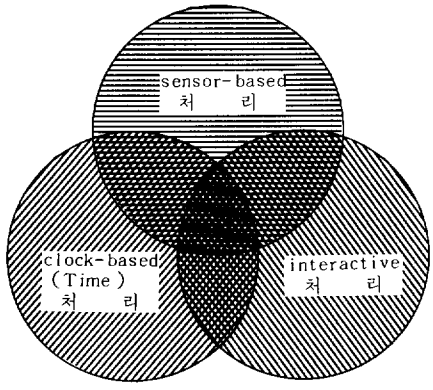

그림 1. On-line 처리

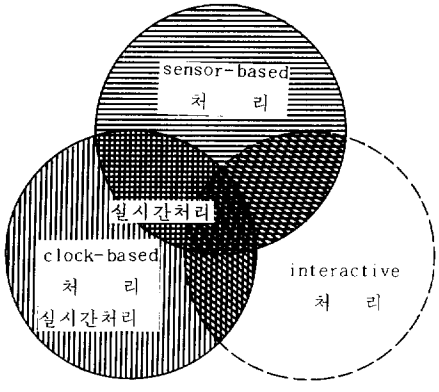

그림 2. 실시간 처리의 정의

실시간 처리는 그림 2와 같이 on-line 처리에서 sensor-based 와 clock-based 처리에 해당하며 약간의 interactive 처리를 포함한다.

실시간 처리는 외부 process 와 접속장치 (예, D/A 또는 A/D 변환기, 등) 를 통하여 연결되어 제한된 시간내에 필요한 서비스를 수행한다. 컴퓨터를 통하여 데이타를 수집할 경우는 외부 process 의 시간간격을 위주로 수행한다. 또한 제어 기능을 수행할 경우는 제한된 시간내에 적절한 응답을 보낸다.

실시간 서비스는 가능한 빠른 처리 보다는 외부 process 가 요구하는 deadline 내에 확실한 처리가 중요하다. 보장된 응답을 위해서는 시간제한을 받지 않는 일반적인 데이타 처리 시스템과는 달리 실시간 응용분야에 적합한 하드웨어, 소프트웨어, language 등을 필요로 한다.

2.2 실시간 시스템의 요구사항

실시간 처리 시스템은 제한된 시간내에 여러가지의 일을 수행할 수 있어야 한다. Critical 한 일은 deadline 을 넘을 경우 시스템 전체에 영향을 줄 수 있으므로, 실시간 시스템은 다음과 같은 특성을 요구한다.

i) 정확성

일반적인 시스템에서는 차후에 에러를 찾아 보상할 수 있으므로 부정확한 응답도 허용된다. 실시간 시스템에서는 부정확한 응답을 허용할 수 없다. 외부 물리적 현상은 대부분 관성을 갖고 있으므로 부정확한 출력에 대한 영향을 쉽게 없앨 수 없다. 그러므로 큰 재난을 야기시킬 경우를 제외하고는 부정확한 출력보다는 응답을 보내지 않는 것이 좋을 경우가 많다.

ii) 신속성

실시간 시스템에서 deadline 을 벗어난 출력은 쓸모가 없으며 때로는 위험을 일으킬 경우도 있다. 그러므로 제한된 시간내에 정확한 출력을 보내도록 해야 한다. 일반적으로 deadline 의 범위는 수 msec 이내의 빠른 응답으로 부터 수 sec 이상의 늦은 응답을 허용하는 경우가 있으므로 이에 맞추어 적합한 시스템을 설계 구성해야 한다.

iii) 신뢰성

실시간 시스템은 고장에 의한 영향이 크기 때문에 높은 신뢰성을 확보하는 설계가 중요하다. 신뢰도의 평가 척도는 안전규준에 의해 규정되지만, 시스템 설계에 있어서는 고장에 의한 서비스 불가능한 시간의 할당을 나타내는 비가동율이 중요시 된다. 비가동율은 하드웨어의 신뢰도, 장치의 2중화 등의 구성방법, 고장이 일어나서 회복할 때 까지의 처리 방법에 의존한다.

iv) 융통성

대부분의 물리적인 시스템은 시간에 따라 많은 영향을 받는다. 물리적인 노후화는 물론 새로운 기술의 발전으로 기존의 시스템이 상대적으로 빠르게 노후화된다. 또한 응용분야가 확장되므로 요구되는 시스템 특성이 다양해져서 이에 적응시키기 위해서는 시스템의 융통성이 중요하다.

융통성을 갖는 시스템을 구성하기 위해서는 하드웨어와 소프트웨어 모두가 module 로 구성되어야 한다. 각각의 module 은 서로간의 interaction 을 가능한 적게하여 새로운 기술을 부분적으로 쉽게 수용할 수 있도록 해야 한다.

이상과 같은 요구사항은 단일의 processor 를 사용한 시스템에서는 모두 만족시키기 어렵다. VLSI 기술과 소프트웨어 기술의 발전으로 최근에는 여러

개의 processor 를 사용한 multiprocessor 시스템과 이를 통신망을 통하여 연결한 분산처리 시스템으로 실시간 처리 시스템의 요구사항을 만족시켜 나가고 있다.

3 실시간 처리 시스템의 구조

3.1 외부와의 접속장치

실시간 처리 시스템은 외부 process 와 접속을 위한 장치를 갖는다. 접속장치는 외부 process 에 따라서 결정되며, 일반적으로 다음과 같은 3가지가 있다.

1) Digital (또는 binary) 신호 : 논리적인 값에 해당하는 '1' 과 '0' 를 가지고 외부와 접속이 이루어진다. Process 의 상태가 반대되는 2가지로 이루어지는 경우 사용된다. Valve 의 개폐, 스위치의 on / off 가 이에 해당된다.

2) Pulse 신호 : Pulse 의 갯수를 통하여 외부 process 의 상태를 체크하고 제어한다. Pulse counter 가 필요하며, 일정한 시간간격에서 pulse 의 갯수로서 상태 결정 및 제어를 수행한다. 스텝모터 제어가 이에 해당한다.

3) Analog 신호 : 컴퓨터가 처리할 수 있는 신호는 digital 신호이므로 외부 신호가 analog 인 경우 A/D 변환기로 상태를 체크하고, D/A 변환기를 통하여 제어신호를 출력한다. 외부 process 의 analog 신호 레벨의 범위가 다양하므로 일정한 수준으로 증폭하여 신호변환을 한다. 온도측정 및 제어와 같은 많은 응용분야가 있다

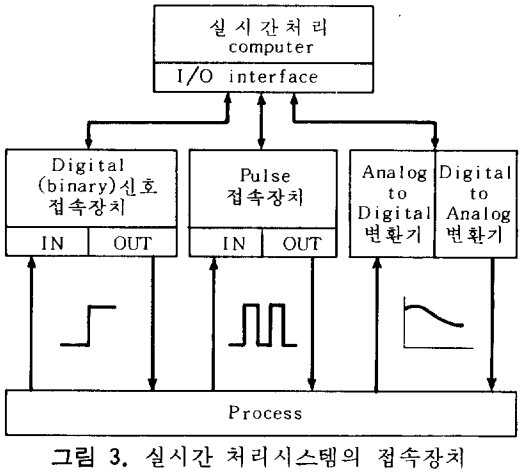

그림 3. 실시간 처리시스템의 접속장치

이와 같은 신호들은 컴퓨터가 처리할 수 있는 신호로 변환된 후에 입출력 접속장치를 통하여 그림 3 과 같이 접속된다.

3.2 컴퓨터 구조

1) Uniprocessor 시스템

초기의 하드웨어 가격이 높을 때 주로 사용되었다. 신뢰도 문제를 해결하기 위해서는 dual processor를 사용해야 하며 확장성 및 융통성이 낮기 때문에 제한된 응용분야를 갖는다.

2) Multiprocessor 시스템

영상처리 시스템과 같이 신속하고 대량의 데이타를 처리해야 하는 응용분야에서는 적합한 특수 용도의 processor 들을 사용하여 deadline 내에 정확한 처리를 수행할 수 있다.

Multiprocessor 시스템은 신뢰도, 정확성, 신속성을 만족시킬 수 있다. 또한 modular 구조로 융통성을 갖도록 하므로서 급격한 기술진보로 인한 시스템의 낙후화에 쉽게 대체할 수 있고, 각종 응용분야에 adaptive 하게 적응시켜 나갈 수 있다.

신뢰도를 향상시키기 위한 dual-processors 시스템 구조는 그림 4 와 같다. 하나의 CPU는 stand-by 상태로 diagnostic 및 프로그램 개발을 위한 서비스를 제공한다. 실시간 처리 CPU에 에러가 발생하면 작업을 넘겨 받아 대신 수행한다.

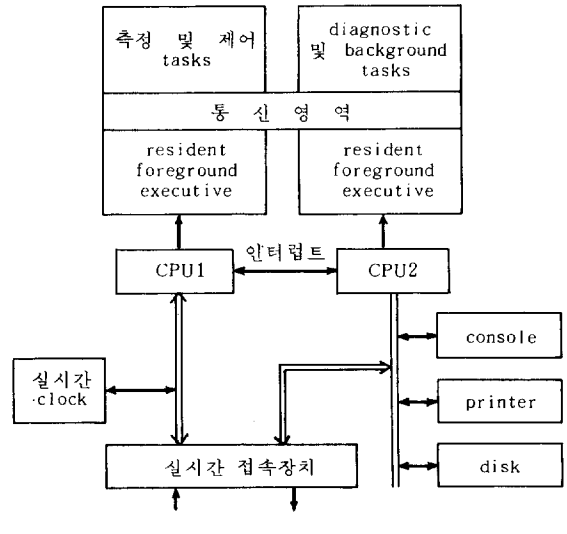

그림 4. Dual－processor 시스템 구조

3.3 분산처리 시스템

공장의 자동화와 같은 시스템은 제어대상이 지역

적으로 분산되어 있으므로 분산시스템을 사용하는 것이 좋다. 하드웨어 가격이 낮아지고 있는 최근에는 분산처리 시스템이 많은 장점을 가질 수 있다.

분산시스템의 구조는 여러가지 있으나 실시간 처리 시스템에서는 신뢰도가 중요한 요소이므로 사무자동화를 위한 국지 통신망에 비하여 선택 범위가 제한된다.

i) 계층적 구조

실시간 응용에서 컴퓨터의 기능이 계층적으로 정해지는 특성에 따라서 피라미드 형태의 구조를 갖는다. 이 구조는 비교적 작은 여러가지의 일을 수행할 때 각각의 일을 전용 컴퓨터들에게 전담시키므로서 시스템을 효율적으로 운용할 수 있다.

전용 컴퓨터를 관리하는 컴퓨터는 전용 컴퓨터에서 처리할 수 없는 일을 도와주며 전체 시스템을 신뢰성있도록 운용한다.

2 level 을 갖는 계층구조는 그림 5와 같으며, 일반적으로 데이타 수집이나 제어 시스템에서 널리 사용된다.

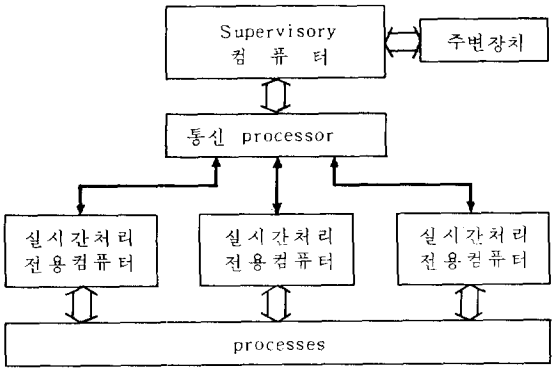

그림 5. Two-level 컴퓨터 계층

ii) bus 형태

하나의 케이블에 여러가지의 기능을 갖는 컴퓨터들이 접속장치를 통하여 연결된다. 이 형태의 특징은 접속장치가 수동소자로 구성되므로 비용이 적게 들고 구성이 간단하며 신뢰도가 높다.

통신망 acess 를 위한 프로토콜로 CSMA (Carrier Sense Multiple Access) 방식이 주로 사용되고 있다. CSMA 방식은 통신망 사용권이 랜덤으로 결정되므로 최대 지연시간을 보장할 수 없다.

Token-passing 방식은 최대 access 지연시간을 보장할 수 있으나 신속성과 신뢰성에서 CSMA방식에 비해 떨어진다.

실시간 시스템에서 access 프로토콜은 fault 의 영향, 긴급 메시지 송수신, deadline 등에 따라서 결정해야 한다.

iii) loop 형태

Loop 형태는 여러개의 컴퓨터들이 loop 모양으로 연결된다. 이 형태는 전송로에서 데이타가 충돌하지 않으므로 access 지연시간을 보장할 수 있다. 또한 잡음의 영향을 받지 않고 대역폭이 넓은 광섬유를 사용할 수 있는 장점이 있다. 그러나 접속장치의 고장 등의 사고에 따라서 통신망 전체에 영향을 줄 수 있으므로 신뢰도가 낮은 단점이 있다.

4 실시간 시스템의 소프트웨어

4.1 Operating System

오늘날 많은 컴퓨터들이 실시간 시스템이라고 한다. 일반적으로 이들이 의미하는 것은 시스템이 외부의 events 에 대하여 빠른 반응을 나타낼 수 있는 능력을 말한다. 이 경우 대부분은 실시간 작업에 높은 priority 를 주어서 서비스를 제공한다. Priority 에 의한 실시간 서비스는 여러가지의 실시간 작업이 수행될 때는 같은 level 에서 수행시키므로 정해진 deadline 내에 서비스를 제공할 수 없는 경우가 생길 수 있다.

실제적으로 실시간 서비스는 외부 events 에 대하여 정해진 deadline 내에 확실히 제공되어야 한다. 매우 빠른 응답을 요구할 경우는 그 응용에 적합한 구조를 갖는 전용의 시스템을 사용해야 한다.

실시간 서비스는 deadline 내에 서비스가 제공되지 못했을 경우 야기되는 문제의 크기에 따라서 실시간 작업수와 prority 를 결정한다. 일반적으로 심각한 문제를 야기시킬 경우 실시간 작업, 심각하지 않은 경우 시분할 작업, 응답시간에 관계없는 경우 background 작업으로 나누어 처리한다. 여기서 실시간 작업인 경우 응답 시간을 확신할 수 있는 작업량 내에서 처리해야 한다.

실시간 처리를 위한 O.S.의 기본적인 구성은 일반적인 데이타 처리를 위한 multiprogramming O.S.에 비하여 다음과 같은 부분에서 특성을 가져야 한다.

1) 실시간 process 들을 위한 scheduler

Deadline 내에 실시간 process 들이 서비스를 제공할 수 있도록 scheduling 이 가능해야 한다.

2) File 의 효율적인 관리

주로 데이타 acquisition 에서 고속으로 많은 데이타를 저장할 수 있도록 contiguous file system을 가져야 한다.

3) process 간의 효율적인 통신

실시간 응용에서 하나의 작업을 효율적으로 수행하기 위하여 병렬처리를 필요로 한다. 이를 위해서 각 process 간의 통신을 원활히 할 수 있는 기능이 필요하다.

4) 메모리 관리

빠른 응답을 위하여 실시간 process 는 swapping 되지 않는 foreground 영역에서 수행되어야 한다. 즉, 실시간 process 는 항상 메모리에 resident 하여야 한다.

5) Device Driver

실시간 시스템 clock, graphic 터미널 등의 실시간 응용분야에 따라 필요한 주변 장치를 관리할 수 있어야 한다.

실시간 O. S. 에서 deadline 내의 보장된 응답 이외에도 신뢰성이 중요하다. 일반적으로 신뢰성과 빠른 응답은 서로 상반된 특성을 가진다. 그러므로 실시간 시스템에서 신뢰성을 높이기 위한 시간적인 손실과 요구되는 응답 시간은 응용 분야에 따라서 적절한 수준에서 구성되어야 한다.

신뢰도를 높이기 위하여 O. S. 는 job 이 수행되는 동안 초기 정보를 비롯하여 restart 에 필요한 모든 정보를 보관하고 수시로 점검해야 한다. 수행하는 도중 에러 (error)가 발생하면 최근의 상태를 통하여 restart 를 할수 있어야 한다. 또한 자원의 재할당에 의하여 유동적으로 고장난 자원을 다룰 수 있어야 한다.

4.2 실시간 언어

초기에는 실시간 program 은 효율적이고 빠른 수행을 위하여 어셈블리 (assembly)언어로 구성되었다. 이는 program 구성이 힘들고 디버깅 (debugging)이 까다롭다. 또한 program 의 관리가 어렵고 readability 가 나쁘다.

High-level 언어는 어셈블리 (assembly) 언어의 많은 단점을 해결할 수 있으나 효율적인 수행이 어렵다.

최근에는 실시간 처리에서 보다 효율적인 수행을 위하여 concurrent 언어가 개발되었다. Concurrent 언어는 mutual exclusion 문제와 같은 어려운 점들이 있으나, 프로그램 구성 및 관리가 쉽고 readability 가 좋으며 실시간 job 을 보다 효율적으로 수행할 수 있다.

실시간 언어가 갖추어야 할 요소들은 다음과 같다.

- O 실시간 응용 프로그램의 scheduling 을 위한 O. S. 와의 interface 기능
- O 빠른 file 관리를 위한 random file 을 handling 할 수 있는 기능
- O Concurrent 처리를 위한 process 간의 원활한 통신 기능
- O 실시간 처리에서 많은 부분을 차지하는 bit-by-bit 처리 기능
- O 실시간 응용에 따른 주변장치들을 관리할 수 있는 다양한 I/O routines
- O 수행 시간을 관리할 수 있는 기능

최근에는 VLSI 의 발전에 힘입어 multiprocessor 나 분산처리 시스템을 실시간 처리에 응용하여 이들이 제공하는 여러가지의 장점을 활용하고 있다. 이에 맞추어 software 개발 도구도 multiprocessor 와 분산처리에 적합한 언어로 발전되었다. 이같은 언어의 발전 상황은 그림 6 과 같이 3단계로 나눌 수 있다.

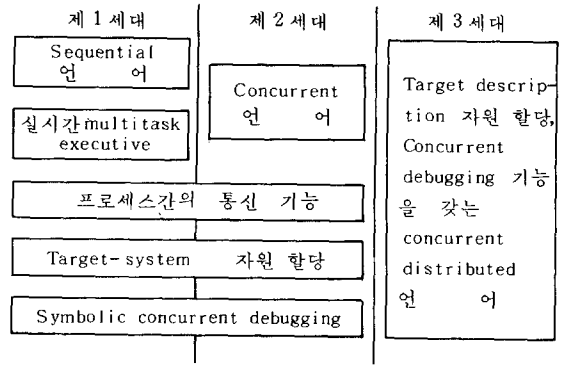

그림 6. 언어의 발전 상황

1 세대 언어는 실시간 O. S. 가 process 간의 통신을 서비스한다. 자원 할당, 시스템 배치 기능과 process 간의 통신 기능이 미약하다.

2 세대 언어는 semaphore, monitor, randezvous 등과 같은 풍부한 process 간의 통신 기능을 갖고 있다. Concurrent 처리가 가능하나 분산처리에 적합한 기능이 부족하다. Concurrent Pascal, Pearl, ADA 와 같은 언어가 여기에 해당한다.

3 세대 언어는 2 세대 언어에서 부족한 분산 처리를 위한 기능들을 갖고 있다. 즉, 자원 할당과 process 간의 통신을 직접 관리할 수 있으며 target 시스템의 배치에 adaptive 하게 적용시킬 수 있다.

5 앞으로의 전망

실시간 처리의 응용 분야는 크게 process 제어 시

스템과 데이타 acquisition 시스템으로 나눌 수 있다. Process 제어 시스템은 process 의 변수들을 센서(sensor)들로 부터 데이타를 받아서 필요한 프로세싱을 한 후 그 결과로 부터 process 를 제어 할 수 있는 신호를 보낸다. 여기에서 중요한 점은 정해진 시간내에 컴퓨터에서 데이타를 처리하여 결과를 보내야 한다. 주로 공장의 자동화, 화학 공정 제어 등에서 많이 볼 수 있다.

데이타 acquisition 시스템은 실시간으로 데이타를 수집하여 처리하되 결과를 process 에 보내지 않고 storage 에 저장한다. 수집된 데이타의 처리는 실시간으로 처리할 경우도 있고, 일단 저장한 후 차후에 처리할 수도 있다. 실시간 처리는 영상 처리에서 많이 응용되며, 고속 처리를 요하는 경우가 많다.

과거에는 하드웨어 가격이 높아 uniprocessor 나 dual-processor 로 실시간 처리 시스템을 구성하였으나, 반도체및 그 주변 장치의 기술 발전으로 인하여 실시간 특성에 적합한 시스템이 사용되고 있다. 일반적으로 이들 전용 시스템은 multiprocessor 로 구성되어 신속성, 신뢰성 및 융통성을 제공한다. 또한 통신망 기술의 발전으로 국지 통신망이 공장의 자동화와 같은 응용 분야에서 널리 사용된다.

최근의 급격한 기술 발전은 기존의 시스템을 쉽게 낙후시키고 있다. 다양한 응용 분야의 확장으로 인하여 계속적으로 새로운 응용 분야에 적응 가능한 구조를 갖는 시스템이 필요하게 되었다. 이와 같은 문제를 해결하기 위하여는 실시간 처리 시스템의 기본적인 구조가 modular 특성을 가져야 한다. 일반적으로 multprocessor 나 분산 처리 시스템이 어느 정도 modular 특성을 가지고 있으나, 급격한 기술과 다양한 응용 분야에 쉽게 적응해 나가기 미흡하다.

Modular 시스템의 적응성, 확장성, 변환성과 같은 특성은 여러가지의 특성을 요구하는 실시간 응용 분야에서 보다 더 유용하다. 이는 하드웨어의 module 화 뿐만 아니라 소프트웨어에서도 module 로 구성되어야 하며 이같은 module 에 적합한 architecture 를 가져야 한다.

이같은 modular 시스템은 multiprocessor 시스템을 기본으로 하여 발전되며 분산 처리 시스템으로 확장될 것이다. 즉, local 시스템은 modular 특성을 갖는 multi-processor 시스템이 되며, 이들이 고속의 통신망을 통하여 연결된 분산처리 시스템이 될 것이다.

참고문헌

1) Saffer, S.I. and Mishelevich, D.J., "A Definition of Real-Time computing," Comm. ACM, 18, 544, Sapt. 1975.

2) Mellichamp, Real-Time Computing, Van Nostrand Reinhold, 1983.

3) A. R. White, Douglos J. Ross, M. Martin Jaylor, "UNIX Support for Guaranteed Real-Time Processing," Summer 83 Toronto Conference Proceedings, Thursday, 14 July 1983.

4) H. Lycklama, C. Christensen, "A Minicomputer Satellite Processor System," B.S.T.J. Vol. 57, No. 6, July-August 1978, pp. 2103-2113.

5) James D. Schoeffler, "Distributed Computer Systems for Industrial Process Control," IEEE Trans. Computer, Feb. 1984.

6) P. Garctti, P. Laface and S. Rivioira, "On the Development of a Distributed Operating System Kernel for Real-Time Applications," IFAC. Real Time Programming, 1981.

7) Maurelio Boari, "Multiple-Microprocessor Programming Techniques : MML, a New Set of Tools," IEEE Trans. Computer, Jan. 1984.

8) H. Lycklama, D. L. Bayer, "The MERT Operating System," B.S.T.J., Vol. 57, No. 6, July-August 1978, pp. 2049-2087.

9) G. W. R. Luderer, J. F. Maranzawo, B. A. Tague, "The UNIX Operating System as a Base for Applications," B.S.T.J., Vol. 57, No. 6, July-August 1978, pp. 2201-2209.

10) David M. Harland, "High Speed Data Aquisition : Running a Real-Time Process and a Time-Shared System (UNIX) Concurrently," Software-Practice and Experience, Vol. 10, pp. 273-281, 1980.

11) David R. Cheriton, Michael A. Malcolm, "Thoth, a Portable Real-Time Operating System," CACM, Vol. 22, No. 2, Feb. 1979.

12) Robert G. Arnold et. al, "A Modular Approach to Real-Time Supersystems," IEEE Trans. Comp., Vol. C-31, No. 5, May 1982.

13) Svetlama P. Kartashev, "Supersystems : Cur-

rent State-of-The-Art Guest Editor's Introduction," IEEE Trans. Comp. Vol. C-31 No. 5, May 1982.

14) T. Moto-Oka, Fifth Generation Computer Systems, North-Holland, 1981.

15) Gérard LE LANN, "On Real-Time Distributed Computing," Information Processing 83, R.E.A. Mason (ed.) IFIP, 1983.

16) Hartwing U. Steusloff, "Advanced Real-Time Languages for Distributed Industrial Process Control," IEEE Trans Computer, Feb. 1984, pp. 37~46.

17) Patrice Humbert-Droz, Hart Jansson, "Micro-Concurrent Pascal suits real-time applications," Electronic Design May 28, 1981.