실시간 소프트웨어 기술

이 직 열
(한국전자통신연구소 제어기기 개발실장)

1. 서 론

컴퓨터 시스템 구현에 소요되는 H/W가격이 S/W에 비해 상대적으로 낮아짐에 따라 군사용에 국한되었던 실시간 시스템 응용이 그 대상을 공정제어, 산업자동화, 의료 및 과학연구, 항공전자, CAT(computer aided testing) 등으로 넓혀지고 있다.

실시간 시스템은 엄밀한 시간제약과 높은 신뢰도를 만족시키며 데이타를 획득하고 처리된 데이타를 이용하여 제어를 수행해야한다. 이러한 시스템은 제약조건이 매우 엄밀한 경우 단일 대상을 위한 전용시스템개발로서 전용 H/W 및 S/W에 의하여 구현되어지나, 일반적인 실시간 시스템은 기존의 H/W에 범용의 실시간 운영체제와 실시간 언어를 이용한 응용프로그램을 내장시켜 구현하게 된다.

실시간 운영체제는 실시간 시스템의 성능을 결정짓는 중요한 요소이며, 실시간 언어는 운영체제에 의하여 특징지어지는 시스템의 성능을 개선하지 못하는 반면 저하시킬 수 있는 요소로서, 이 두가지 S/W에 의하여 시스템의 성능이 대부분 결정되어진다.

실시간 시스템의 성능을 나타내는 중요한 요소는 응답시간과 입·출력 데이타 전송비로서, 응답시간은 context switching과 interrupt latency시간에 의하여 결정되어진다. Context switching이란 실시간 운영체제에서 가장 기본이 되는 multi-task 처리를 위한 task 사이의 전환을 의미하며, 이는 수행중인 task의 정보를 기억시키기 위한 시간과 scheduling과 같은 부가적인 기능을 처리하기 위한 시간이 소요되게 되어, context switching이 너무 빈번하게 발생되는 경우 전체 시스템의 성능을 저하하게 되므로 cycle time을 최적으로 유지해야 한다. Interrupt latency시간은 interrupt 신호발생시 이에 대응하여 처리해야 할 task를 context switch시키기 이전까지 요구되는 시간을 나타내며, 이 외에도 응답시간에 영향을 미치는 요소로는 계산속도와 기억장치에의 접근시간이 있다. 이들 요소들은 복합적으로 상호작용을 일으키며, 제조회사에서 제공하는 특성들은 다른 요소들을 제외한 상황에서 측정된 데이타로서, 이에 의하여 실제의 시스템의 성능을 특징짓기는 곤란하다.

데이타 전송비는 시스템 입·출력시 직렬 또는 병렬 형태의 아날로그 또는 디지탈 신호에 대한 이송속도를

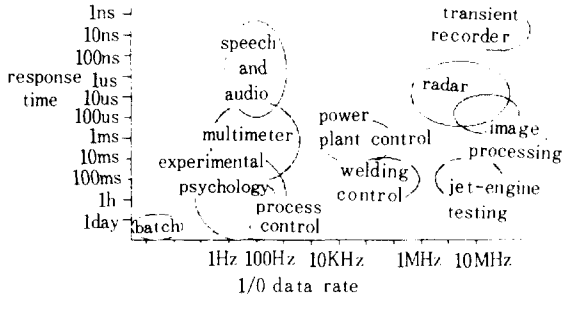

그림 1. 실시간 시스템의 응용분야

표 1. 실시간 운영체제의 현황

Company	System Name	Processors	Size(KB)	Speed(μs)	User	Language
Advanced Micro Devices	OS Engine	Z8002	8/16	40-400	1	None
Hemenway Corp	MSP	68000/Z8002	8/24	N/A	1	B. P. Fort
Hewlett Packard Co.	RTE-A	A600	30/50	55-140	6-12	B. F. P
Industrial Programming Inc.	MTOS/68-KF	68000	-/16	20-500	16	None
Intel Corp.	iRMX 86	iAPX86/88, iAPX286	0.5/13	30	4	B, C, F, P Cob, PL/M
Motorola Inc.	VERSAdos	68000	12/60	200	varies	F, P
System & Software	REX-80	8086/8087, 8088	4/-	100- 350	1	C, P, PL/M
Masscomp	Real-Time UNIX	Dual 68000s	28/64		any	C, F, P
Hunter & Ready	VRTX	Z80, 8086, 68000, Z8000	4/	100	-	C, P, PL/M

(주) 1. Size(kernel / O S)
2. B=Basic, C=C Language, F=Fortran, P=Pascal, Cob=Cobol

나타내는 요소로서, 그림 1에 응답시간과 전송비를 응용분야에 대하여 개략적으로 나타내었다.

실시간 시스템 구현시 S/W 설계자는 앞서 기술한 요소들 외에 고장진단, 회복등과 같이 우발적인 사고에 대처할 수 있는 기능을 시스템내에 포함시켜야 한다. 실시간 시스템의 기능 중 중요한 것중의 하나는 interrupt 처리기능이다. 일반적으로 여러 interrupt 신호는 그 중요성을 고려한 우선순위를 지정하여, 중요한 interrupt는 다른 interrupt 발생에 관계없이 처리되어 시스템의 시간제약조건을 만족시키도록 한다. Interrupt 당한 task가 제대로 재처리되기 위해서는 수행중인 task의 모든 정보를 저장한 후 우선순위가 높은 interrupt task를 깨우게 된다. 이와같은 처리시 deadlock이나 endless loop를 피하도록 해야한다. Interrupt처리시 시스템의 엄밀한 시간제약을 만족시키기 위한 방안으로 dynamic calculation에 의한 scheduling을 들 수 있는데, 이는 사건발생율과 처리시간을 이용하여 계산하게 된다.

본 소고에서는 실시간 시스템 구현에 소요되는 실시간 운영체제 및 실시간 언어에 대한 개략적인 사항을 기술한 후, 실제적인 응용 프로그램의 예를 들어 실시간 시스템의 이해를 돕도록 하였다.

2. 실시간 운영체제

실시간 시스템이란 다양한 외부의 사건 또는 공정에 대하여 엄밀한 시간제약조건을 만족시키며 상호작용함으로서 원하는 기능을 수행토록 하는 시스템으로 특징지을 수 있다. 다양한 기능을 동시에 처리하기 위해서는 응용 프로그램을 여러개의 모듈로 나눈다음 이들 모듈을 독립적인 task로 구동시키는 방법이 이용된다. 실시간 운영체제는 응용 프로그램이 이와같은 실시간 환경하에서 처리되어지도록 그 환경을 보장하여야 한다. Multi-task 처리에 의하여 얻어질 수 있는 효과로는 한정적인 자원, 예를들면, CPU를 효율적으로 이용할 수 있는점과 실시간 시스템의 응용 프로그램 개발시 상호작용하는 복합적인 기능을 여러개의 독립적인 task로 나눔으로서 S/W개발을 용이하게 하는 점을 들 수 있다.

Multi-task 처리기능과 더불어 실시간 운영체제가 가져야 할 기능으로 interrupt 처리를 들 수 있다. 실시간 시스템은 다른 시스템에 비해 입력 channel의 수가 많은 것이 특징이며, 각 channel을 통하여 인지되는 외부사건 또한 비동기적으로 발생하며, 발생사건에 대한 중요성과 처리속도도 다르게 된다. 앞서 기술한 기능외에 데이타 보호, task간의 통신 및 동기화등이 실시간 운영체제가 가져야 할 기능이다.

그림 2는 실시간 운영체제를 구현하기 위한 방안을

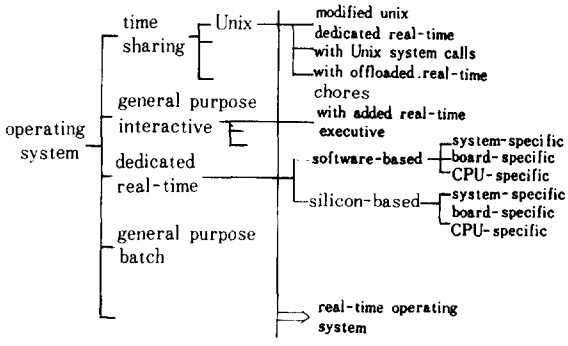

그림 2. 실시간 운영체제의 분류

나타낸 것이다.

그림 2에서 보여주는 바와같이 실시간 운영체제의 구현방안은 Unix를 이용하는 방법, 운영체제에 실시간 기능을 부가하는 방법, 전용 실시간 운영체제를 개발하는 방법으로[1] 나눌 수 있다. Unix를 이용하는 방법은 C언어가 시스템 프로그래밍에 용이한 점, C언어에 대한 개발환경이 풍부한 점, 기존 Unix가 다양한 S/W 패키지를 가지고 있다는 점이 커다란 장점으로 판단되어 이 방법이 널리 이용되고 있다. 실시간 기능을 부가하는 방법은 소요되는 실시간 기능을 run-time library로 구현하여[2] 기존의 운영체제 위에서 수행되므로, 응용 프로그램의 입장에서 보면 하나의 계층이 덧붙여진 형태로 구성되어지므로 시스템의 성능이 다른 구현방법에 비해 낮을 수 있다. 마지막으로, 전용 실시간 운영체제는 그림 1의 여러 응용분야 중에서 응답속도와 데이타 전송비가 큰 분야의 시스템 개발에 이용된다.

실시간 시스템을 위한 S/W개발의 다른 한 방법으로는 실시간 모니터 프로그램 구현을 들 수 있다. 이는 소형 μ-processor 응용 시스템 또는 embedded system에 이용되는 방법으로서 기존의 모니터 프로그램에 실시간 기능을 덧붙임으로서 소규모 실시간 시스템을 효율적으로 개발할 수 있는 방법이 되고있다.

이 장의 뒷부분에서는 실시간 운영체제가 지녀야하는 실시간 기능을 구현하기 위한 여러가지 기법에 대하여 개략적으로 기술한다.

1) Task 관리

Task 관리기능은 scheduling, interrupt task 처리, 시간관리로 나누어진다. 여기서 시간관리란 주기적인 scheduling을 위한 time quantum, 각 task에 대한 CPU 할당, 주기적으로 수행되어져야 할 task의 개시와 같은 기능등을 포함한다. 이와같은 기능을 가진 모듈의 특성은 trigger mechanism, scheduling방법 및 preemptive/non-preemptive방식에 의하여 결정되어진다. Trigger mechanism이란 scheduling을 위한 신호를 발생시키는 방식으로서, 이는 interrupt 구동방식, task 구동방식 및 이들의 혼합방식으로 구분된다. Interrupt 구동방식은 task간의 context switching이 H/W 또는 S/W interrupt에 의존하며, task 구동방식은 수행중인 task 관리를 위한 task의 지시에 따라 다른 task를 제어하는 방법이다. 혼합방식은 이들 두 방법을 결합한 것으로, context switching시 수행중인 task의 CPU status, program counter 및 각종 register의 내용을 각 task에 할당된 TCB(task control block)에 저장하여야 한다.

Scheduling을 위하여 여러가지 방법이[3] 이용되고 있으나 산업용 실시간 시스템에서는 우선순위 방식과 round-robin방식이 널리 이용되고 있다. 그림 3은 두가지 방식을 그림으로 나타낸 것이다.

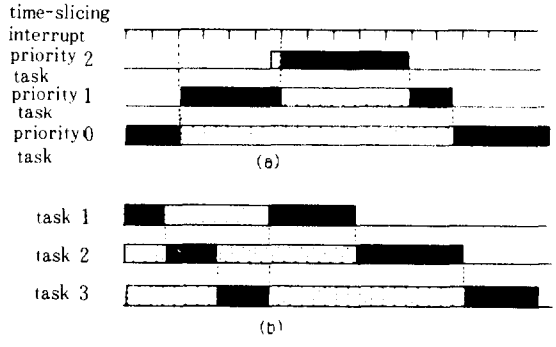

그림 3. Priority vs. round-robin scheme

Non-preemptive방식을 적용한 scheduler는 수행중인 task가 스스로 CPU를 포기하지 않는 한 지속적으로 수행하게 되며, preemptive방식은 수행중인 task보다 우선순위가 높은 task가 깨워지는 경우 그 자신은 suspend 되어진 후 scheduling을 기다리게 된다.

Scheduler가 효율적으로 여러 task들을 관리하기 위하여서는 TCB가 이용되며 그림 4에 한 예를 나타냈다. 여기서 time-out counter는 그 task가 CPU를 할당 받은 시각과 시간관련 queue에 등록된 task의 시간제어에 각각 이용된다. Program counter는 수행이 지

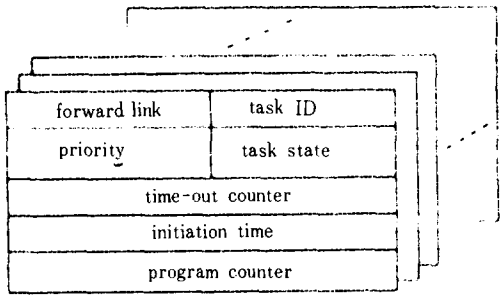

그림 4. Task control block

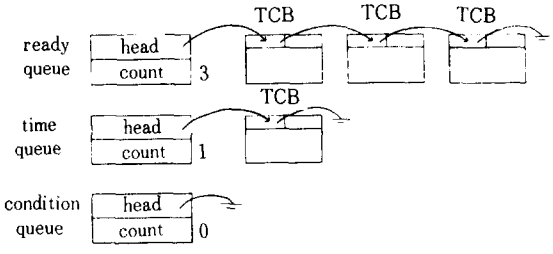

그림 5. Task queue의 예

연되는 경우 다음 수행할 번지를 기억시키기 위한 장소이다.

Forward link는 동일한 상태에있는 task를 하나의 queue에 등록함으로서 scheduler로 하여금 효율적으로 관리할 수 있도록 한 것으로 그림 5에 그 예를 나타내었다.

여기서 ready queue는 ready state에 있는 task를, time queue는 시간에 관련되어 수행되지 않는 task를, condition queue는 각종 조건, 예를들면, 다른 task로부터 수행하라는 신호를 수신하기까지는 수행될 수 없는 task를 위한 queue이다. 예를 들어, 공정의 상태를 주기적으로 감시하는 task의 반복 주기가 500mSec인 경우, 이 task는 생성시 반복주기 값이 TCB의 time-out counter에 기록되어진 후 time queue에 등록된다. 주기적인 time-slicing interrupt(예, 10mSec)에 의하여 time-out counter값이 감소되어 영에 이르게 되면, 우선적으로 CPU를 할당받으며 그 시각을 TCB의 initiation time에 기록하며, 후로는 initiation time과 time-out counter값을 이용하여 주기적으로 반복동작하게 된다. 이러한 방법에 의하면 task내에서 소요되는 수행시간을 응용 프로그램 작성시 고려치 않더라도 반복주기를 엄밀하게 만족시킬 수 있게 된다.

2) Task간의 통신 및 동기화

실시간 시스템에서 task는 독립된 프로그램이지만 task들 사이에는 데이타의 공유나 교환의 필요성이 있다. Task간의 데이타 교환은 메시지를 이용하여 구현되어지며, 주로 dynamic buffer와 공유 메모리를 이용한 방법이 널리 이용되고 있다. Dynamic buffer 방법에서, 임의의 task가 메시지 전송을 원할 경우, 실시간 운영체제 중 통신처리 루틴은 사용되지 않는 버퍼중 일부를 배당하여 그 곳에 메시지를 기록한 후 버퍼의 포인터 값만을 수신측의 task에 전달하며, 수신 task는 송신측에 수신완료를 통보하게 된다. 만일 task간에 동기화가 필요한 경우, 이와같은 메시지 전송방법을 이용할 수 있다.

전절의 task관리와 관련하여 좀더 자세히 설명하면 다음과 같다. 두 task가 데이타를 송·수신하고자 하는 경우 수신측 task의 요청에 의하여 실시간 운영체제는 버퍼가 데이타로 채워져 있는지 확인한 후 데이타가 있는 경우 수신측에 버퍼에 대한 포인터 값을 넘겨주는 반면, 빈 경우에는 요청한 task를 상태천이시켜 condition queue에 머무르게 한다. 만일 요청한 task가 데이타 수신까지의 시간을 지정하는 경우 그 시간을 경과하게 되면 데이타 요청은 무시되며 다음 동작을 진행하게 된다. 무조건적으로 데이타 수신을 원하는 경우, condition queue에 머무르게 되며, 송신측 task에서 데이타를 넘겨주게 되면 condition queue에서 ready queue로 옮겨져 scheduling을 기다리게 된다.

공유 메모리를 이용한 방법에서는, 통신을 원하는task들이 접근 가능토록 공유 메모리를 지정한 후 이를 통하여 데이타 교환을 수행한다. 일반적으로 공유 메모리는 세부분의 영역으로 나누어진다. 하나는 송. 수신 데이타의 타입을, 다른 하나는 수신측 task로 하여금 수신후 취하여야 할 행동을, 마지막 영역은 데이타 저장 또는 데이타에 대한 포인터 값을 기록하는데 이용된다. 각각의 버퍼는 특정의 송.수신 task를 위하여 지정되어지므로 dynamic buffer방식보다는 많은 메모리 영역을 필요로 하게된다.

그림 6은 task간의 통신을 도식적으로 나타낸 것이다.

3) 기타 기능

앞서 기술한 기능외에 실시간 시스템이 가져야할 기능으로는 interrupt처리와 mutual exclusion이 있으며, 이들 기능들은 일괄처리용 운영체제와 동일한 방법으로

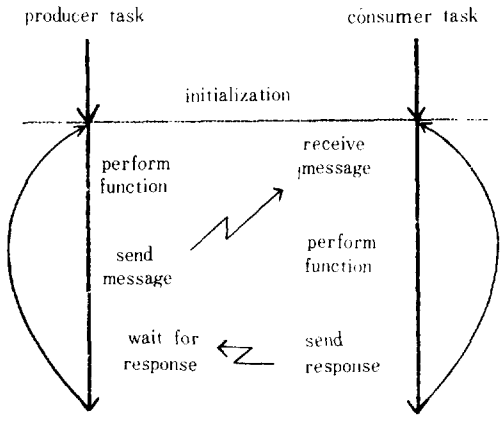

그림 6. task간의 통신

구현되어지나,[4] 응용 프로그램 작성시 사용자들의 편의성을 위하여 다양한 system call이 제공되어진다.

비동기적으로 발생하는 interrupt를 처리하기 위한 방안중의 하나는 다음과 같다. Interrut 신호는 interrup vector table에 기록된 번지에 의하여 interrupt handler를 구동시킨다. Interrupt handler 구동시 모든 interrupt는 불능상태에 있게되며, 처리시간이 짧은 경우 handler에 의하여 처리되어지나, 소요시간이 긴 경우는 interrupt task에 의하여 처리되도록 한다. Interrupt task는 고유의 우선순위를 가지며 scheduler에 의하여 다른 task와 동일하게 처리되어진다.

비동기적으로 병행처리되는 여러 task들이 공통의 데이타를 접근하거나, 처리루틴의 일부가 외부의 interrupt나 우선순위가 높은 task의 방해없이 수행되어져야할 필요성이 있게 된다. 예를 들면, 주기적으로 입력된 데이타를 처리하는 동안에 다른 데이타가 입력되어 처리되어진다면 처리된 데이타는 그 의미를 잃게 된다. 이와 같은 현상을 방지하기 위하여 실시간 운영체제는 mutual exclusion 기능을 가져야 한다. 이의 구현 방안으로는 guard flag나 binary semaphore를 이용하거나, 수행중인 task의 우선순위를 일시적으로 높이는 방법도 이용되고 있다.

3. 실시간 언어

실시간 시스템의 구현방법은 크게 두가지로 나누어진다. 하나는 실시간 기능을 제공하는 실시간 운영체제와 실시간 환경하에서 응용프로그램 작성을 용이하게 해주는 실시간 언어를 동시에 이용하는 방법으로서, 이는 시스템의 기능이 다양하고 주변장치에 대한 제약이 거의 없는 비교적 큰 시스템개발에 적합한 방법이다. 다른 하나는 실시간 운영체제를 도입하지 않고 기본적인 실시간 기능을 가진 실시간 언어만을 이용하는 방법으로,[5] 이는 보조기억 장치나 콘솔등 주변장치가 필요치 않는 소규모 시스템 또는 큰 장비의 일부로서 몇개의 장치를 제어하기 위한 실시간 제어기 등의 개발에 적합하다.

실시간 운영체제와 동시에 이용되는 실시간 언어는 주로 데이타 처리를 용이하게 해주는 특징을 가진 반면, 실시간 운용체제를 이용치 않는 실시간 언어는 실시간 운영체제가 갖는 아래와 같은 기능을 가져야한다.

- Multi-task 처리
- 실시간 기능의 손쉬운 구현
- 입·출력 장치에의 직접적인 접근

Multi-task 처리 방법은 전장의 실시간 운영체제의 경우와 동일하다. 두번째 특징을 갖는 예로는 SCHEDULE, SIGNAL, WAIT 등과 같은 명령어를 들 수 있다. NASA에서 우주왕복선 개발등에 이용한 HAL / S 언어의 경우

SCHEDULE THIS JOB AT 300 SECONDS

라는 문장에 의하여 임의의 시간 또는 외부사건에 의하여 schedule하도록 할 수 있다.

Intel의 PL/M 언어의 경우

OUTPUT(address) = 0AH

라는 문장에 의하여 원하는 장치로 임의의 데이타를 직접 출력시킬 수 있다.

이와 같은 특징외에 프로그래밍의 용이성이 요구된다. 이는 실시간 시스템의 프로그램이 일괄처리용 프로그램에 비해 상대적으로 매우 복잡하기 때문에 더욱 요구된다.

Multi-task하에서 응용프로그램을 효과적으로 작성하기 위해서는 실시간 운영체제와 마찬가지로 task 간의 통신이 가능하여야 한다. 구현방법으로는 semaphore, mailbox, message system이 이용되며, 이는 실시간 운영체제와 동일하다. Semaphore는 데이타의 전송이 없는 사건의 알림에, maibox는 실질적인 데이타 전송에, message system은 mailbox와 동일하나, mailbox의 경우 데이타의 저장장소를 나타내는 포인터 값만을

표 2. 실시간 언어의 현황

언 어	개 발 기 관	시 스 템
Basic 09	Microware systems Corp.	6809 systems
MacBasic	Analog Devices Inc.	CP/M
Fortran 5	Data General Corp.	AOS, AOS/VS
SSS-Fortran	Supersoft Inc.	CP/M, CP/M-86, MSDOS
Micropower/Pascal	Dtgital Equipment Corp.	RT-11
Microprocessor/Pascal	Texas Instruments Inc.	Ii-990, VAX, IBM 370
Pascal	Advanced Digital Products	UCSD p-System
Pascal	Omegasoft Inc.	6809 systems
Pascal	Oregon Software	PDP-11, VAX 68000
Ada	Supersoft Inc.	CP/M
micro Concurrent Pascal	Enertek Inc.	8086/88, Z8000, 68000
Modula-2	Villtion Systems	UCSD p-System MSDOS
Path Pascal	Intelkenasc Inc.	68000-based system
6809 C Language	Microware System Corp.	6809 systems, PDP-11

넘겨주게 되나, 여기서는 실질적인 데이타 이동이 일어나므로 사용자측에 프로그래밍의 용이성은 보장되나 데이타 이동에 따른 성능저하 문제가 야기된다.

4. 실시간 S/W 구현 예

본 장에서는 앞서 기술한 내용의 이해를 돕기위하여 "공정정보처리 시스템 개발(1985. 7~1986. 7)" 과제에서[6] 저자가 개발한 RTEX(real-time executive) 를 간략하게 소개하며, 이를 이용한 응용 프로그램을 기술한다.

RTEX는 실시간 운영체제가 소요되지 않는 소규모 제어기에 실시간 기능을 제공할 수 있는 아래와같은 기능을 갖도록 하였다.

- multi-task처리
- task간의 통신
- 엄밀한 시간 제어
- mutual exclusion
- H/W, S/W interrupt 처리
- priority based preemption scheduling

상기와 같은 기능을 만족시키기 위하여 RTEX는 표 3과 같은 22개의 system call로 구현되었다.

그림7은 500mSec주기로 데이타를 획득한 후 이를 데이타 처리 task에 mailbox를 이용하여 전송하는 기능을 구현한 예이다. 여기서 사용된 언어는 Intel의 PL/M

표 3. summary of system calls

task control	mailbox control	region control
CREATE-TASK	CREATE-MBX	CREATE-REGION
DELETE-TASK	DELETE-MBX	DELETE-REGION
GET-PRIORITY	RECEIVE-MESSAGE	ENTER-REGION
RESUME-TASK	SEND-MESSAGE	EXIT-REGION
SUSPEND-TASK		
SLEEP-TASK		
CYCLE-TASK		
time control	interrupt control	memory control
GET-TIME	CATALOG-INTERRUPT	CREATE-MEMORY
SET-TIME	ENTER-INTERRUPT	DELETE-MEMORY
	EXIT-INTERRUPT	

이다.

5. 맺음말

실시간 시스템 구현에 필요한 실시간 운영체제와 실시간 언어에 대한 기본적인 특징과 이를 위한 구현방안 및 현황을 소개하였다. 실시간 운영체제에 대해서는 task관리, task간의 통신 및 부수적인 기능을, 실시간 언어는 실시간 운영체제를 요하지 않는 실시간 시스템 개발에 이용되는 언어를 중심으로 기술하였다.

현재, 국내의 실시간 시스템의 응용분야로는 국방, 통신 및 공장자동화 분야가 주를 이루고 있으며, 공장자

```
sample-program:ddo;
  task1 : procedure public;
      obj-ptr = CREATE-MEMORY(4, except);
      do forever;
        call CYCLE-TASK(task-id1, 500, except);

        call SEND-MESSAGE(mbx1, obj-ptr, except);
      end;
  end task1;
  task2 : procedure public;
      do forever;
        obj-ptr = RECEIVE-MESSAGE(mbx1, except);
          .
      end;
  end task2;
  task-id1 = CREATE-TASK(10, task1, except);
  task-id2 = CREATE-TASK(20, task2, except);
  mbx1 = CREATE-MBX(except);
end sample-program;
```

그림 7. 응용 프로그램의 예

동화 업계에서는 독자적인 시스템개발 보다는 외국의 시스템을 OEM 방식으로 도입하여 응용 프로그램을 추가하는 형식을 취하고 있다.

제조업계의 전반적인 경향이 제조공정의 자동화로 치닫게 됨에 따라 이에 소요되는 실시간 시스템의 수요가 크게 확대되리라 예측되어진다.

참 고 문 헌

1) K. H. Sears, et al, "Software concurrency in real-time control system: a software nucleus," Software-Practice and Experience, VOL. 15, Aug., 1985.

2) D. A. Crowl, "A real-time Fortran executive", IEEE MICRO, Aug., 1985.

3) E. A. Parrish, et al, "A scheduler for real-time task control in microcomputer," IEEE Trans. IECI, NO. 1, 1978.

4) J. L. Peterson, et al, Operating system concepts, Addison-Wesley Publishing Company, 1985.

5) M. Schindler, "Real-time languages speak to control applications," Electronic Design, July, 1983.

6) 한국전자통신연구소, "공정 정보 처리 시스템 개발," 과학기술처, 1986.