(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.[6]
G06F 9/46

(11) 공개번호 특1998-0010769
(43) 공개일자 1998년04월30일

(21) 출원번호 특1997-0031230
(22) 출원일자 1997년07월05일
(30) 우선권주장 96-177898 1996년07월08일 일본(JP)
(71) 출원인 미쯔비시 덴끼 가부시끼가이샤 기따오까 다까시
일본국 도꾜도 지요다꾸 마루노우찌 2쪼메 2-3
(72) 발명자 구로사와 하사요시
일본국 도꾜도 지요다꾸 마루노우찌 2쪼메 2-3 미쯔비시 덴끼 가부시끼가이샤 내
(74) 대리인 장수길, 이상희, 구영창

심사청구 : 있음

(54) 리얼 타임 제어 시스템

요약

리얼 타임성을 보증하지 않는 퍼스널 컴퓨터용 오퍼레이팅 시스템(PC-OS)에서, 입출력 장치 드라이버의 위치부여로 리얼 타임 처리를 제공함으로써, PC-OS의 변경을 수반하지 않고, 리얼 타임 어플리케이션과 PC 어플리케이션의 공존을 실현한다.

PC-OS가 비출력 디바이스 드라이버를 호출하는 기구, 획득한 CPU 사용권을 각 리얼 타임 프로세스에 분배하는 제2 스케줄링 수단, 리얼 타임 프로세스로서의 로딩 수단, 및 프로세스간 통신 기구를 구비한다.

대표도

도1

명세서

[발명의 명칭]

리얼 타임 제어 시스템

[도면의 간단한 설명]

도1은 본 발명의 제1 실시예를 도시한 시스템 구성도.

도2는 본 발명의 제1 실시예에서의 PC-OS의 인터럽트 처리를 도시한 흐름도.

도3은 본 발명의 제1 실시예에서의 PC-OS의 디바이스 드라이버 입출력 요구처리를 도시한 흐름도.

도4는 본 발명의 제1 실시예에서 RT 프로세스가 리얼 타임 제어 기구의 루틴으로서 실장된 경우에서의 제2 스케줄링 수단의 처리를 도시한 흐름도.

도5는 본 발명의 제1 실시예에서 RT 프로세스가 독자적인 콘텍스트를 가진 경우의 제2 스케줄링 수단의 처리를 도시한 흐름도.

도6은 본 발명의 제1 실시예에서의 RT 프로세스간 통신 수단의 처리를 도시한 흐름도.

도7은 본 발명의 제1 실시예에서 PC 프로세스가 데이터를 수신하는 경우의 PC 프로세스간 통신 수단의 처리를 도시한 흐름도.

도8은 본 발명의 제1 실시예에서 PC 프로세스가 데이터 송신하는 경우의 PC 프로세스간 통신 수단의 처리를 도시한 흐름도.

도9는 본 발명의 제2 실시예를 도시한 시스템 구성도.

도10은 본 발명의 제2 실시예에서의 사용권 감시 수단의 처리를 도시한 흐름도.

도11은 본 발명의 제2 실시예에서의 제2 스케줄링 수단의 처리를 도시한 플로 차트.

도12는 본 발명의 제2 실시예에서의 사용권 감시 수단의 다른 처리예를 도시한 흐름도.

도13은 본 발명의 제3 실시예를 도시한 시스템 구성도.

도14는 본 발명의 제3 실시예에서의 경합 제어 수단의 처리를 도시한 흐름도.

도15는 종래의 리얼 타임 제어 방식을 도시한 시스템 구성도.

도16은 종래의 리얼 타임 제어 방식을 도시한 다른 시스템 구성도.

* 도면의 주요부분에 대한 부호의 설명

101 : 제2 스케줄링 수단 102 : RT 프로세스

103 : RT 프로세스간 통신 수단 104 : PC 프로세스간 통신 수단

105 : RT 로드 수단 106 : PC-OS 보호 수단

107 : PC 리얼 타임 제어 기구 108 : 제1 스케줄링 수단

109 : 드라이버 조립 기구 110 : PC-OS

111 : PC 프로세스 901 : 사용권 감시 수단

1101 : 경합 제어 수단

[발명의 상세한 설명]

[발명이 속하는 기술분야 및 그 분야의 종래기술]

본 발명은, 리얼 타임성을 요구하는 프로세스와, 범용 퍼스널 컴퓨터상에서 동작하는 프로세스와의 공존을 목적으로 한, PC 리얼 타임 제어 기구에 관한 것이다.

최근, 퍼스널 컴퓨터(이하, "PS"라 함)의 급속한 보급에 따라, 종래 폐쇄적이라고 했던 리얼 타임 시스템에서도, PC상에서 유통되고 있는 많은 소프트웨어 자산을 활용할 것이 요구되어 왔다.

PC상의 소프트웨어 자산을 활용하기 위해서는, 리얼 타임 시스템상에서 동작하는 RT 프로세스와, PC상에서 동작하는 PC 프로세스와의 협조 동작이 필요하며, 종래, 다음의 두 가지 방법이 취해지고 있었다.

[종래예 1]

제1 방법으로서는, 예를 들면 「계측과 제어 제34권 제3호」 (1995년 3월 발간) 201쪽에 기재된 것이 있다.

이 방법은 도 15에 도시한 바와 같이, PC-OS(110)가 주행하는 GPU1(1301)과 리얼 타임 OS(1304)가 주행하는 CPU2(1302)를 따로따로 탑재하고, RT 프로세스(102)는 리얼 타임 OS(1304) 제어하에서, PC 프로세스(111)는 PC-OS(110) 제어하에서 동작시키도록 한 것이다.

RT 프로세스와 PC 프로세스는 CPU1 및 CPU2로부터 액세스가능한 공유 메모리(1303)를 기재하여 데이터의 교환을 행하고 있다.

[종래예 2]

제2 방법으로서는, 예를 들면 「인터페이스 1956년 6월호」 (CQ 출판) 142쪽에 기재된 것이 있다.

이 방법은 도 6에 도시한 바와 같이, 리얼 타임 OS를 코아로서. 그 위에 에뮬레이터형으로 PC-OS를 탑재한다.

RT 프로세스(102)는 직접 리얼 타임 OS가 제공하는 서비스를 이용하고, PC프로세스(111)는 PC-OS 에뮬레이터 상에서 동작하도록 되어 있다.

종래의 PC 리얼 타임 제어 기구는 이상과 같이 구성되어 있었기 때문에, 리얼 타임 OS 및 RT프로세스가 동작하는 GPU나 공유 메모리 등의 여분의 하드웨어를 탑재할 필요가 있고, 비용이 높아진다고 하는 문제점이 있었다.

또한, PC-OS 에뮬레이터로서 구성하는 경우에는, 여분의 하드웨어에 의한 비용 증가는 억제될 수 있으나, PC-OS의 버전업때마다 PC-OS 에뮬레이터 전체를 변경할 필요가 있고, 소프트웨어 개발비의 증가를 초래한다는 문제점이 있었다

[발명이 이루고자 하는 기술적 과제]

본 발명은 이러한 문제점을 해결하기 위한 것으로서, RT 프로세스와 PC 프로세스가 공존하는 시스템을 실현하는데 있어서, 하드웨어 비용을 억제함과 동시에, PC-OS의 버전업에 대하여도 유연하게 대응할 수 있는 PC 리얼 타임 제어 기구를 제공하는 것을 목적으로 한다.

본 발명에 관한 제1 리얼 타임 제어 시스템은, 어플리케이션 프로세스에 대하여 CPU리소스의 할당 및 스케줄링을 행하는 제1 스케줄링 수단, 및 디바이스 장치에 대하여 입출력 처리를 행하는 디바이스 드라이버를 조립하는 드라이버 조립기구를 구비한 오퍼레이팅 시스템에 있어서, 디바이스 장치로부터의 인터럽트에 대하여 리얼 타임 응답성이 요구되는 처리를 상기 오퍼레이팅 시스템으로부터 디바이스 드라이버로서 액세스되는 리얼 타임 제어 기구로서 구성하며, 상기 리얼 타임 제어 기구는 상기 디바이스 장치에 대한 처리 대응에 구성된 리얼 타임 프로세스, 및 상기 리얼 타임 프로세스에 대하여 상기 CPU 리소스의 할당 및 스케줄링을 행하는 제2 스케줄링 수단을 구비하도록 한 것이다.

본 발명에 관한 제2 리얼 타임 제어 시스템은, 제1 리얼 타임 제어 시스템에서, 리얼 타임 제어 기구가 리얼 타임 프로세스간에서의 데이터의 송수신을 행하는 리얼 타임 프로세스간 통신 수단을 구비하도록

한 것이다.

본 발명에 관한 제3 리얼 타임 제어 시스템은, 제1 리얼 타임 제어 시스템에서, 리얼 타임 제어 기구가 오퍼레이팅 시스템상에서 동작하는 어플리케이션 프로세스와 리얼 타임 프로세스간의 데이트 송수신을 행하는 프로세스간 통신 수단을 구비하도록 한 것이다.

본 발명에 관한 제4 리얼 타임 제어시스템은 제1 리얼 타임 제어 시스템에서, 리얼 타임 제어 기구가 오퍼레이팅 시스템 상에서 동작하는 어플리케이션 프로세스를 리얼타임 프로세스로서 등록하는 리얼 타임 프로세스 로드 기구를 구비하도록 한 것이다.

본 발명에 관한 제5 리얼 타임 제어 시스템은, 제1 리얼 타임 제어 시스템에서, 리얼타임 제어 기구가 오퍼레이팅 시스템이 디바이스 드라이버에 대하여 제공하는 서비스 기능을 리얼 타임 프로세스로부터 이용가능하게 하는 오퍼레이팅 시스템 보호 수단을 설치하도록 한 것이다.

본 발명에 관한 제6 리얼 타임 제어 시스템은, 제1 리얼 타임 제어 시스템에서, 리얼 타임 제어 기구는 상기 제2 스케줄링 수단이 CPU 리소스를 오퍼레이팅 시스템으로부터 점유하고 있는 시간을 계측·유지하고, 상기 계측 결과에 기초하여 CPU 리소스를 강제적으로 오퍼레이팅 시스템으로 되돌리는 사용권 감시 수단을 구비하도록 한 것이다.

본 발명에 관한 제7 리얼 타임 제어 시스템은, 제1 내지 제6 리얼 타임 제어 시스템에서, 상기 리얼 타임 제어 기구가 복수인 경우, 상기 복수의 리얼 타임 제어 기구간의 주종 관계 또는 우선 순위 관계를 조정하는 경합 제어 수단을 구비하도록 한 것이다.

[발명의 구성 및 작용]

[제1 실시예]

본 발명의 제1 실시예에 대하여, 도1 내지 도8에 기초하여 설명한다.

도1은 제1 실시예에서의 시스템 구성을 도시한 도면이다.

도1에서, 도면 부호(101)은 CPU 이외의 디바이스 장치로부터의 인터럽트에 대하여, PC-OS가 호출하기전에 등록해 두었던 인터럽트 서비스 루틴에서, CPU 사용권을 RT 프로세스(102a, 102b)에 분배함과 동시에, RT 프로세스의 실행 상태를 관리하는 제2 스케줄링 수단이다.

도면 부호(103)은 RT 프로세스(102a, 102b)간에서의 데이터 교환이나 동기처리를 제공하는 RT 프로세스간 통신 수단, 도면 부호(104)는 RT 프로세스와 PC 프로세스간에서의 데이터 교환이나 동기 처리 기능을 제공하는 PC 프로세스간 통신 수단이다.

도면 부호(105)는 PC 프로세스로서 작성·디버그한 프로세스를 RT 프로세스로서 실행하기 위한 RT 로드 수단이다. 도면 부호(106)은 종래 PC-OS가 디바이스 드라이버에 대하여 제공되고 있는 PC-OS기능을 RT 프로세스로부터 이용하게 하기 위한 PC-OS 보호 수단이다.

도면 부호(107)은 PC 리얼 타임 제어 기구를 도시하며, PC-OS로부터는 1개의 디바이스 드라이버로서 보이며, 내부에 도면 부호(101-106)의 각 수단 및 복수의 RT 프로세스를 갖는다. 도면 부호(108)은 PC-OS가 내부에 갖는, PC 프로세스의 실행 순서를 제어하는 제1 스케줄링 수단, 도면 부호(109)는 디바이스 드라이버를 PC-OS에 조립하기 위한 틀을 제공하는 드라이버 조립 기구, 도면 부호(110)은 PC-OS를 도시한다.

도면 부호(111a-c)는 PC-OS상에서 동작하는 어플리케이션 프로그램인 PC 프로세스이다.

다음에, 동작에 대하여 (1) PC 리얼 타임 제어 기구, (2) PC 프로세스의 입출력 요구에 따른 드라이버의 동작, (3) RT 프로세스를 리얼 타임 제어 기구의 1개의 루틴으로서 제어하는 경우의 제2 스케줄링 수단, (4) RT 프로세스가 독자적인 콘텍스트(context)를 갖고 실장되는 경우의 제2 스케줄링 수단, (5) RT 프로세스간 통신 처리, (6) PC 프로세스간 통신 처리, (7) RT 로드 수단, (8) PC-OS 보호 수단의 순서로 설명해 간다.

(1) 우선, PC 리얼 타임 제어 기구(107)가 어떻게 실행되는지에 대하여 설명한다.

PC 리얼 타임 제어 기구(107)는 PC-OS(110)으로부터 보면 1개의 디바이스 드라이버이며, 디바이스 드라이버의 실행 계기는 CPU외의 디바이스 장치로부터의 인터럽트와 PC 프로세스로부터의 입출력 요구의 2종류가 있다.

도2는 통상의 OS에서의 인터럽트 발생으로부터 대응하는 디바이스 드라이버의 처리가 완료하기까지의 처리의 흐름을 도시하고 있다.

인터럽트가 발생하면, CPU는 PC-OS(110)가 OS의 초기화시에, CPU에 대하여 등록해 둔 루틴을 기동한다. 기동된 OS의 루틴에서는, 우선 단계(201)에서 인터럽트가 발생했을 때의 시스템의 상태를 보존한다.

다음으로, 단계(202)에서 발생한 인터럽트에 대응한 처리 루틴(이하, "인터럽트 서비스 루틴"이라 함)이 사전에 등록되어 있는지를 체크한다. 단계(202)에서 인터럽트 서비스 루틴이 등록되어 있지 않으면 단계(203)으로 진행하고, 예정외의 인터럽트 발생으로서 시스템을 정지하던지, 아무것도 하지않고 단계(205)에 진행하여, 시스템 상태를 복귀하여 처리를 완료한다. 상기 어느 처리를 행하는지는 PC-OS의 종류에 따라 다르지만, 본 발명에는 직접 관여하고 있지 않으므로 여기서는 설명을 생략한다.

단계(202)에서 인터럽트 서비스 루틴이 등록되어 있었으면, 단계(204)에서 상기 인터럽트 서비스 루틴을 호출한다. 인터럽트 서비스 루틴의 처리가 완료되면, 단계(205)에 진행하며, 단계(201)에서 보존한 시스템의 상태를 복귀하고, 인터럽트 처리를 완료한다.

단계(204)에서 호출된 인터럽트 서비스 루틴에서는, 인터럽트를 발생한 디바이스 장치와 데이터를 교환(데이터 입력의 인터럽트인 경우는 디바이스 장치로부터 데이터를 꺼내고, 데이터 출력 완료의 인터럽트인 경우는 출력 데이터를 디바이스 장치에 송출한다)하고, 그 결과 입출력 요구 대기의 PC 프로세스가 존재했다면, 그 PC 프로세스의 대기 상태를 해제하고, 제1 스케줄링 수단(108)을 호출하며, PC 프로세스의 재 스케줄링을 행한다.

또한, 통상의 OS에서는 인터럽트 서비스 루틴으로부터 직접 스케줄링 수단을 호출하면 시스템 처리에 모순을 일으키므로, 인터럽트 처리 완료후에 제1 스케줄링수단(108)을 호출할 수 있는 기구를 갖고 있다.

(2) 다음에, PC 프로세스(111)의 입출력 요구에 따른 디바이스 드라이버의 실행 처리에 대하여, 도 3의 흐름도에 기초하여 설명한다.

PC 프로세스로부터 입출력 요구가 이루어진면, PC-OS(110)의 일부인 드라이버 조립 기구가 호출된다. 드라이버 조립 기구로는 PC 프로세스로부터의 입출력 요구의 종류에 대응한 디바이스 드라이버의 요구 접수 루틴을 기동한다 「단계(301)」. 이 요구 접수 루틴은 예를 들면 데이터의 판독, 기입, 디바이스 장치의 직접 제어 등으로 나누어지며, 디바이스 드라이버의 초기화 처리에서 드라이버 조립 기구에 등록되는 것이다. 디바이스 드라이버의 요구 접수 루틴에서는, 요구 내용에 따라 디바이스 장치와의 데이터 교환을 하지만, CPU의 속도에 비하여 디바이스 장치는 대단히 저속이므로, 드라이버 조립 기구에서는 PC 프로세스를 입출력 완료 대기 상태로 한다 「단계(302)」.

다음으로, 단계(303)에서는 단계(302)에서 PC 프로세스가 대기 상태가 되었기 때문에, 다음에 실행할 PC 프로세스를 선택하기 위하여, 제1 스케줄링 수단(108)을 호출하고, 입출력 요구의 처리를 완료한다.

입출력 완료 대기 상태가 된 PC 프로세스는 도 2에 도시한 인터럽트 서비스 루틴에 의하여 대기 상태를 해제하게 된다.

이상 설명한 바와 같이, PC-OS가 갖는 제1 스케줄링 수단(108)은 디바이스 드라이버의 처리중에는 실행되지 않으며, 그 결과 디바이스 드라이버는 PC 프로세스(111a-c) 또는 제1 스케줄링 수단(108)에 대하여 우선적으로 처리되게 된다.

(3) 다음에, RT 프로세스의 실행을 제어하는 제2 스케줄링 수단(101)의 동작을 도 4에 기초하여 설명한다.

도4는 RT 프로세스가 PC 리얼 타임 제어 기구의 1개의 루틴으로서 실장되는 경우의 제2 스케줄링 수단(101)의 처리를 나타낸 흐름도이다.

제2 스케줄이 수단(101)은 도2의 단계(204)로부터 호출되면, 우선 단계 (401)가 실행되며, PC 리얼 타임 제어 기구 실행중을 나타내는 플래그를 체크하고, 만일 플래그가 "ON"이면 아무것도 하지 않고 처리를 종료하고, 도2의 단계(205)로 되돌아간다.

단계(401)에서, 만일 플래그가 "OFF"이면 단계(402)에서 플래그를 "ON"으로 하여 단계(403)으로 진행하고, 실행 가능한 RT 프로세스의 유무를 체크한다. 체크 방법으로서는, 예를 들면 RT 프로세스의 선두 어드레스(본 실시예의 경우 루틴의 선두 어드레스)를 테이블에 유지하고, 테이블에 유지되어 있는 RT 프로세스는 모두 실행 가능 상태라고 판단하면 된다. 또한, 디바이스 장치로부터의 인터럽트가 주기적으로 발생하는 것이면, 상기 테이블에 카운터를 설치하면, RT 프로세스마다 다른 주기 시간에서의 기동도 가능해진다.

단계(403)에서, 만일 실행 가능 상태의 RT 프로세스가 존재하면 단계(404)에 진행하며, 테이블에 등록되어 있는 루틴을 호출한다. 호출된 루틴(RT 프로세스)에서는, RT 프로세스 특유의 처리를 행한 후 리턴함으로써, 단계(404)로 되돌아가고, 그 후 단계(403)으로 진행한다. 단계(403) 및 단계(404)를 반복하여 실행함으로써 테이블에 등록된 실행 가능한 모든 RT 프로세스가 처리되면 단계(405)로 진행하고, 플래그를 0으로 세트하여 처리를 종료한다.

이 경우, RT 프로세스는 도면에 도시한 바와 같이, 루틴으로부터의 리턴으로 처리를 종료한다.

(4) 다음에, RT 프로세스가 PC 리얼 타임 제어 기구의 1개의 루틴이 아니며, 독자적인 콘텍스트를 갖고 실장되는 경우의 제2 스케줄링 수단(101)의 처리에 대하에, 도 5의 흐름도에 기초하여 설명한다.

이 경우, RT 프로세스는 도 5에 도시한 바와 같이, 프로세스 자체의 초기화 처리부와, 프로세스 특유의 처리를 반복하여(처리하는 형태가 된다. 반복 처리의 종료는 Wait 콜에 의하여 스스로 실행권을 방기함으로써 행한다.

제2 스케줄링 수단(101)은 도 2의 단계(204)로부터 호출되면, 우선 단계(401) 및 단계(402)를 실행한다. 이상의 단계는 도 4의 해당 단계와 동일하다.

다음으로, 단계(403)에서 실행 가능 상태의 RT프로세스를 체크한다. 체크 방법은 도 4의 흐름도를 설명한 경우와 동일하다. 단계(403)에서, 만일 실행 가능 상태의 프로세스가 존재했다면, 단계(501)에 진행하고 단계(403)에서 얻어진 프로세스의 상태 정보를 복귀한다. 이 처리에 의하여 RT 프로세스는 단계(503)에 도시한 개소로부터 다시 실행되지만, RT 프로세스의 처리를 여기서 설명한다.

RT 프로세스는 최초로 기동되면, 자기 자신의 초기화 처리를 행하고 「단계(502)」, 단계(503)으로 진행한다. RT 프로세스의 최초의 기동은, 디바이스 드라이버인 PC 리얼 타임 제어 기구의 초기화 처리에서 호출되게 하면 되고, 또한 PC 프로세스로부터의 디바이스 개시 의뢰 중에서 호출하여도 좋다.

단계(503)에서는 RT 프로세스에 고유의 처리를 행한 후, 단계(504)에서 Wait를 콜한다.

Wait 콜은 PC 리얼 타임 제어 기구가 RT 프로세스에 대하여 제공하는 기능으로서, RT 프로세스의 실행을 일시 중단하는 것이다. 이 기능은 본 발명에서는 필수의 기능이 아니지만, 본 실시예의 설명을 용이하게

하기 위한 것으로서, 호출된 RT 프로세스를 실행 가능 상태로부터 제외하고 「단계(505)」, 상기프로세스의 상태 정보를 유지 「단계(506)」한 후, 다시 제2 스케줄링 수단(101)을 호출한다 「단계(403)」.

단계(501)에서의 RT 프로세스의 상태 정보의 복귀는 상기 단계(506)에서 유지된 상태 정보의 복귀이며, RT 프로세스는 스텝(503)으로부터 실행을 재개한다.

이상의 단계(403), 단계(501)로부터 스텝(506)이 반복되며, 모든 실행 가능상태의 RT 프로세스가 Wait의 콜에 의하여 실행 가능 상태에서 벗어나면, 단계(403)에서는 실행 가능 상태의 프로세스가 없는 상태로 판정되고, 단계(405)에서 플래그를 'OFF'로 하여 PC-OS의 인터럽트 처리 「도 2의 흐름도의 단계(204)」로 되돌아간다.

(5) 다음에, RT 프로세스간 통신 수단의 처리에 대하여 설명한다.

도4 및 도5의 흐름도에서는, RT 프로세스는 각각 독립된 처리로서 설명하였으나, 일반적인 프로세스는 서로 처리의 동기를 취하거나, 데이터의 교환을 행하면서 처리를 수행해간다.

이하에서는, 2개의 RT 프로세스(102a, 102b)에서 가장 간단한 데이터 교환을 행하는 경우를 예를 들어 처리의 흐름을 설명한다.

도6은 2개의 RT 프로세스(102a, 102b)에서, RT 프로세스(102a)가 데이터의 수신을 또한 RT 프로세스(102b)가 데이터의 송신을 하고 있는 도면이다.

우선, RT 프로세스(102a)가 단계(601)에서 데이터의 수신을 행하면, RT 프로세스간 통신 수단의 단계(602)가 호출되며, 데이터의 유무를 체크한다. 데이터가 존재하면 그 데이터를 RT 프로세스의 데이터 수신의 호출시에 지정한 메모리 영역에 카피하고 「단계(603)」, 데이터 수신 처리를 종료하며, RT 프로세스와 단계(504a)에서 Wait를 호출하고, 실행 가능 상태로부터 벗어난다.

단계(602)에서 데이터가 없으면 단계(604)에서 RT 프로세스(102a)를 실행 가능 상태로부터 벗어나게 하고, 단계(605)에서 RT 프로세스의 상태 정보를 유지하여, 단계(606)에서 제2 스케줄링 수단(101)을 호출한다. 여기서 호출되는 제2 스케줄링 수단은, 상기 Wait의 처리에서 설명한 경우와 동일하며, 도 5의 스텝(403)이다.

제2 스케줄링 수단(101)의 처리 결과, RT 프로세스(102b)가 선택되며 상태정보가 복귀되면, RT 프로세스(102b)는 Wait의 직후로부터 실행이 재개되며, 단계(607)에 진행하며 데이터 송신을 호출한다. 데이터 송신이 호출되면, RT 프로세스간 통신 수단의 단계(608)가 실행되며, 데이터 대기 상태의 프로세스의 유무를 체크한다.

만일 데이터 대기 상태의 RT 프로세스가 존재하지 않으면, 단계(609)에서 데이터를 RT 프로세스간 통신 수단이 갖는 메모리 영역에 카피하고, 데이터의 송신을 완료하며, RT 프로세스(102b)에 되돌아가 Wait에 의해 실행 가능 상태로부터 벗어난다. 본 설명에서는 RT 프로세스(102a)가 단계(604)에서 데이터 대기 상태가 되고 있으므로, 단계(608)에서는 데이터 대기 프로세스라고 판정되며, 단계(610)으로 진행한다.

단계(610)에서는 데이터를 RT 프로세스의 지정된 메모리 영역에 카피하고, 단계(611)에서 RT 프로세스를 실행 가능 상태로 한다. 이로써 데이터 송신 처리는 완료하고, RT 프로세스(102b)에 되돌아가 단계(504b)에서 Wait을 호출하며, RT 프로세스(102b)는 실행 가능 상태로부터 제외된다.

그 후, 도 5의 흐름도의 단계(403)에서는 실행 가능 상태의 프로세스로서 RT 프로세스(102a)가 선택되며, 단계(501)에서 상태가 복귀되면 단계(606)에 되돌아온다.

이로써, RT 프로세스(102a)의 데이터 수신 처리는 완료되고, 단계(504a)에서 Wait가 불려지면, 실행가능 상태로부터 벗어나고, 제2 스케줄링 수단(101)에 의해, 실행 가능 상태의 프로세스가 없기 때문에 PC-OS의 인터럽트 처리로 되돌아간다.

(6) 다음에 PC 프로세스간 통신 수단의 처리에 대하여 도 7 및 도 8에 기초하여 설명한다.

PC-OS등 통상의 계산기 시스템에서는 PC 프로세스끼리, 또는 RT 프로세스끼리 서로 처리의 동기를 취하거나, 데이터의 교환을 행하면 되지만, PC 프로세스와 RT 프로세스가 존재할 경우는 PC 프로세스와 RT 프로세스간에서도 데이터의 교환동기가 필요해진다.

이하에서는, PC 프로세스와 RT 프로세스에서 가장 간단한 데이터 교환을 행하는 경우를 예로 들어, 각각에 대하여 처리의 흐름을 설정한다. 도 7은 PC 프로세스가 데이터의 수신을, RT 프로세스가 데이터 송신을 행하는 경우의 도면이다.

또한, 도8은 PC 프로세스가 데이터의 송신을, RT 프로세스가 데이터의 수신을 행하는 경우의 도면이다.

실제로는 PC 프로세스에서는 데이터의 송신은 PC 리얼 타임 제어 기구에 해당하는 디바이스 드라이버에 대한 데이터 출력 요구, 데이터 수신은 데이터 입력요구에 대응하고 있고, PC 프로세스에 대하여는 그 형식에서 기능을 제공해도 좋으며, PC 프로세스에 대하여 데이터 출력 요구를 데이터 송신에, 데이터 입력 요구를 데이터 수신에 대응시키는 라이브러리를 제공하여도 좋다.

우선, PC 프로세스가 데이터의 수신을 RT 프로세스가 데이터의 송신을 행하는 경우의 동작을 도7에 대하여 설명한다.

제1 스케줄링 수단(108)에 의하여 PC 프로세스가 실행되며 단계(701)에서 데이터 수신을 요구하면, 드라이버 조립 기구에 의하여 데이터 입력 요구로서 PC 리얼 타임 제어 기구의 PC 프로세스간 통신 수단이 호출된다.

PC 프로세스간 통신 수단에서는, 우선 단계(702)에서 데이터의 유무를 조사한다. 만일 데이터가 존재하면 단계(703)으로 진행하고, 통상의 디바이스 드라이버와 마찬가지로 데이터 입력 요구 완료로서 드라이

버 조립 기구에 제어를 되돌리고, 그 결과, 데이터 수신 완료로서 PC 프로세스가 단계(701)에 이어서 실행된다.

단계(702)에서, 만일 데이터가 없으면 단계(704)에 진행하고, PC 프로세스를 디바이스로부터의 데이터 입력 대기 상태로서, PC-OS에 통지한다.

다음에, 단계(705)로 진행하며 데이터 입력 요구 완료로서 드라이버 조립 기구(109)로 제어를 되돌린다. 드라이버 조립 기구에서는 PC 프로세스는 디바이스 입력 대기 상태가 되고 있으므로, PC-OS의 제1 스케줄링 수단(108)을 호출하고, 디바이스 입력 대기 상태의 PC 프로세스 이외의 PC 프로세스를 실행하게 된다.

PC-OS로의 디바이스 인터럽트로서 제2 스케줄링 수단(101)이 호출되며, 그 결과 RT 프로세스가 실행되면, PC 프로세스간 통신 수단의 단계(706)가 호출되고 데이터 수신 대기 프로세스의 유무를 체크한다.

단계(706)에서 만일 데이터 수신 대기의 프로세스가 없으면, 단계(707)에서 데이터를 유지하고 RT 프로세스로 되돌아간다. 본 실시예의 경우는 PC 프로세스가 데이터 수신 대기이므로 단계(708)로 진행하며, PC 프로세스가 데이터 입력 요구시에 지정한 PC 프로세스의 메모리 영역에 데이터를 카피하고, 단계(709)에서 PC 프로세스의 데이터 입력 대기 상태가 해제된 것을 PC-OS에 통지한다.

PC-OS에서는 PC 프로세스를 입력 대기로부터 실행 가능 상태로 하지만, 도2의 흐름도의 설명에서 상술한 바와 같이, PC-OS로부터는 디바이스 인터럽트 처리중으로 인식되고 있으므로, 제1 스케줄링 수단(108)은 호출되지 않고, 단계(709)에 제어를 되돌린다. 이로써 PC 프로세스간 통신 수단의 데이터 송신 처리는 종료하고, RT 프로세스에 제어를 되돌린다.

실행 가능한 모든 RT 프로세스가 처리되면 제2 스케줄링 수단(101)으로부터 PC-OS의 인터럽트 처리로 되돌아가고, PC-OS는 도2의 흐름도의 설명에서 상술한 바와 같이, 인터럽트 처리내에서 디바이스 입력 대기가 해제된 PC 프로세스가 존재하므로, 제1 스케줄링 수단(108)이 호출되며, (경우에 따라) PC 프로세스가 데이터 수신 완료의 형으로 실행이 재개된다.

다음에, PC 프로세스가 데이터의 송신을 RT 프로세스가 데이터 수신을 행하는 경우에 대하여 도8에 기초하여 설명한다.

디바이스로부터의 인터럽트로부터 제2 스케줄링 수단(101)이 호출되며, 그 결과 RT 프로세스가 실행되며, 데이터 수신이 호출되면 PC 프로세스간 통신 수단이 호출된다.

PC 프로세스간 통신 수단에서는, 우선 단계(801)에서 데이터의 유무를 체크하여, 만일 데이터가 있으면 RT 프로세스의 지정된 메모리 영역에 데이터를 카피하고 「단계(802)」, 데이터 수신을 종료하며, RT 프로세스에 제어를 되돌린다.

단계(801)에서, 만일 데이터가 없으면 단계(803)에서 RT 프로세스를 데이터 수신 대기로서 실행 가능 상태로부터 벗어나게 하고, 상태 정보를 유지한 「단계(804)」 후, 제2 스케줄링 수단(101)을 호출한다 「단계(805)」. 이때 호출되는 제2 스케줄링 수단은 Wait의 설명을 한 경우와 동일하게, 도5의 단계(403)이다.

데이터 수신대기가 된 RT 프로세스 이외의 실행 가능 상태의 RT 프로세스가 모두 실행되면, 제2 스케줄링 수단(101)에 의해 PC-OS의 인터럽트 처리로 되돌아가고, 디바이스로부터의 인터럽트 처리가 완료된다. 그 결과 PC-OS의 제1 스케줄링수단(108)이 호출된다.

또한, 오퍼레이팅 시스템에 의해서는, 디바이스로부터의 인터럽트 처리가 완료하면, (디바이스 드라이버 처리에서 디바이스 입력 대기의 프로세스가 1개도 대기 상태가 해제되지 않은 경우에는) 인터럽트가 발생한 지점에 되돌아가는 것이 있다. 많은 경우에는 어느 하나의 PC 프로세스가 실행중이며, 인터럽트 처리 완료에 따라 처리가 재개되지만, PC-OS는 각 PC 프로세스에 CPU를 평등하게 할당하도록 하기 위하여, 어느 것은 제1 스케줄링 수단(108)에 의해 PC 프로세스가 실행되게 된다.

PC 프로세스에서는, 단계(806)에서 데이터 송신을 요구하면, 드라이버 조립기구에 의해 데이터 출력 요구로서 PC 리얼 타임 제어 기구의 PC 프로세스간 통신 수단이 호출된다. PC 프로세스간 통신 수단에서는, 우선 단계(807)에서 데이터 대기 상태의 RT 프로세스의 유무를 체크한다. 만일 데이터 대기 상태의 RT 프로세스가 존재하지 않으면, 단계(808)에서 데이터를 PC 프로세스간 통신 수단이 가진 메모리 영역에 카피한다.

다음에, 단계(809)에서 데이터 출력 요구 완료를 드라이버 조립 기구에 통지하고 처리를 종료한다. 그 결과, PC 프로세스에는 데이터 송신 완료로서 실행이 재개된다. 본 실시예에서는 RT 프로세스가 데이터 대기 상태이므로, 단계(807)로 부터는 단계(810)으로 진행하고, 데이터를 RT 프로세스가 지정한 메모리 영역에 카피한다.

다음에, 단계(811)에서 RT 프로세스를 실행 가능 상태로 하고, 단계(809)에 진행하며 데이터 출력 요구 완료를 드라이버 조립 기구에 통지하여 처리를 완료한다. 단계(811)에서 실행 가능 상태가 된 RT 프로세스는, 다음의 디바이스 드라이버로부터의 인터럽트에 의해 제2 스케줄링 수단(101)에 실행되며, RT 프로세스의 데이터 수신 처리가 완료한다.

또한, 단계(811)에서 RT 프로세스를 실행 가능 상태로 한 후, 데이터 출력 요구 완료를 통지하고 있지만, RT 프로세스로의 응답성을 높이기 위하여, 제2 스케줄링 수단(101)을 호출하고, 데이터 출력 요구 완료의 통지 전에 RT 프로세스를 실행하도록 하여도 좋다. 이 경우, RT 프로세스가 호출하는 Wait에서 데이터 출력 요구 완료를 통지하고나서 처리를 종료하지 않으면 안된다.

(7) 다음에, RT 로드 수단(105)의 처리에 대하여 설명한다.

일반적인 계산기 시스템에서는 디바이스 드라이버의 개발은, 어플리케이션의 개발에 비하여 주의 깊게

행할 필요가 있고, 어려움이 있다. 본 발명의 PC 리얼 타임 제어 기구에서는 RT 프로세스를 디바이스 드라이버의 내부의 루틴으로서 실행하기 위하여, 종래는 RT-OS의 어플리케이션으로서 개발되고 있던 RT 프로세스에 비하여, 개발에 어려움을 수반한다.

그 때문에, RT 로드 수단(105)은 RC 프로세스(111c)로서 개발·디버그한 어플리케이션을 디바이스 드라이버인 PC 리얼 타임 제어 기구의 RT 프로세스(102b)로서 조립하는 기구이다.

RT 로드 수단(105)은 RT 프로세스를 PC 프로세스로서 작성·디버그하는 경우의 제한에 강하게 의존한다. 예를 들면 PC 프로세스의 1개의 루틴으로서 작성된 RT 프로세스를 RT 로드 수단에 의해 RT 프로세스로서 등록 요구하는 경우에는, RT 로드 수단은 지정된 루틴의 개시 어드레스로부터 루틴의 사이즈 만큼, 디바이스 드라이버인 PC 리얼 타임 제어 기구 내부의 메모리 영역에 카피하고, 도4의 흐름도의 설명에서 상술한 테이블에 루틴의 개시 어드레스를 등록하면 된다.

PC 프로세스로서 작성·디버그가 완료한 PC 프로세스 전체를 RT 프로세스로서 등록하는 경우에는, RT 로드 수단은 예를 들면 외부 기억 장치상에 격납된 PC 프로세스를 PC 리얼 타임 제어 기구내의 메모리 영역에 읽어들이고, 그 선두 어드레스를 도4의 흐름도의 설명에서 상술한 테이블에 등록하면 된다.

이상의 RT 로드 수단에서는, RT 프로세스로서 작성·디버그된 루틴 또는 PC 프로세스가 PC-OS가 제공하는 기능(시스템 콜 등)을 이용하는 경우에는, PC 리얼 타임 제어 기구내에서 동일한 기능을 제공하지 않으면 안되는 것은 당연하다. 또한, 이들 기능은 PC 프로세스로부터는 단순한 서브 루틴으로서 프로그램되므로, PC 리얼 타임 제어 기구내에서는 그들 어드레스를 RT 프로세스로서 로드 또는 실행하면서 해결할 필요가 있는 것도 당연하다.

이 어드레스 해결 방법으로서는, 예를 들면 PC 프로세스를 디버그 종료 후, 서브 루틴만 어드레스 미해결 상태로 재 컴파일하고, 그것을 RT 로드 수단으로 프로그램을 카피하면서 미해결 어드레스를 PC 리얼 타임 제어 기구 내부의 대응하는 서브 루틴에 치환해 가는 방법이 고려된다.

이외에도 공지의 방법이 몇 가지 있으나, 본 발명에서는 어떠한 방법을 사용해도 좋다. 또한, RT 프로세스로서 PC 프로세스를 작성하는 경우의 사용 제한으로서 PC-OS가 제공하는 기능은 사용 금지로 하여도 좋다.

(8) 다음에 PC-OS 보호 수단(106)에 대하여 설명한다.

드라이버 조립 기구를 갖는 PC-OS는 디바이스 드라이버로부터 이용 가능한 PC-OS의 기능을 제공하고 있다. 이들 기능은 PC-OS가 PC 프로세스에 제공되고 있는 기능과는 통상은 상이한 것이다. 또한, 디바이스 드라이버는 PC-OS의 일부로서 동작하는 경우가 많고, RT 프로세스로부터 안이하게 드라이버로의 제공 기능을 사용하는 것은 시스템의 정지에 이어지는 중대한 문제를 초래하는 경우가 있다.

PC-OS 보호 수단은 PC-OS가 드라이버에 제공하는 기능을 보다 안전하게 RT 프로세스에 제공하기 위한 변환 루틴이며, 엄밀한 에러 체크나 액세스할 어드레스의 체크, PC-OS의 동작에 모순을 일으키는 처리의 배제 등을 행한다.

[제2 실시예]

다음에, 본 발명의 제2 실시예를 도9 내지 도12에 기초하여 설명한다.

도면 부호(901)는 PC 리얼 타임 제어 기구에서 CPU를 점유하고 있는 시간을 감시하는 사용권 감시 수단이다. PC 리얼 타임 제어 기구는 디바이스 드라이버로서 등작하므로, 제1 스케줄링 수단의 스케줄링 대상으로는 되지 않는다. 그 때문에, 다수의 RT 프로세스가 동작하면 제1 스케줄링 수단에 제어가 전달되지 않으며, PC 프로세스가 장시간 동작할 수 없는 상태가 발생한다.

사용권 감시 수단은 PC 리얼 타임 제어 기구가 CPU를 점유하고 있는 시간을 감시, 계상(計上)하고, 적절한 조건하에 PC-OS에 제어를 강제적으로 되돌리고, PC 리얼 타임 제어 기구와 PC-OS로 CPU를 적절히 분배할 수 있도록 하는 것이다.

또한, 도면에서 도면 부호(101-111)는 도1에 도시할 바와 동일하다.

다음에, 동작에 대하여 설명한다.

사용권 감시 수단은, 도2의 단계(204)로부터 호출되는 인터럽트 서비스 루틴으로서 동작한다.

우선, 단계(1001)에서 PC 리얼 타임 제어 기구가 동작 중인지 여부를 판별한다. 판별 방법은 도4의 단계(401)과 동일한 방법이 바람직하다.

만일 동작중이면, 단계(1002)에서 PC 리얼 타임 제어 기구가 동작하고 있는 시간을 계상한다. 계상 방법은 간단한 카운터이어도 좋고, 동작 시간의 누계이어도 좋다.

다음으로, 단계(1003)에서, 미리 정해진 조건을 단계(1002)에서 계산한 정보가 만족하고 있는지 여부를 판정한다. 미리 정해진 조건이라는 것은, 예를 들면 PC 리얼 타임 제어 기구가 연속적으로 CPU를 점유하고 있는 상한치이어도 좋고, 일정 시간내에서의 CPU의 점유율의 상한치이어도 좋다.

단계(1003)에서, 조건을 만족하고 있는, 즉 CPU의 사용권을 PC 리얼 타임 제어 기구로부터 PC-OS에 강제적으로 되돌리는 상태라고 판단되면, 단계(1004)로 진행하며, 강제 종료 플래그를 세트하여 종료한다.

이 강제 종료 플래그는 제2 스케줄링 수단에서 체크되지만, 이 체크 처리가 추가된 제2 스케줄링 수단의 처리 흐름은 도11에 도시한다.

도11은 도4와는 단계(404)의 다음에 단계(1101)의 강제 종료 플래그가 추가되어 있는 점 만이 다르다.

단계(1101)에서 강제 종료 플래그가 세트되어 있는 경우는, 예를 들어 실행가능 상태의 RT 프로세스가

존재하고 있어도, RT 프로세스로의 CPU분배를 중지하고, PC-OS에 제어를 되돌리게 된다.

다음에, 단계(1003)에서, 조건을 만족하고 있지 않은 경우는, 그대로 사용권 감시 수단의 처리를 중지하고(PC 리얼 타임 제어 기구가 동작중이므로), RT 프로세스의 처리가 인터럽트가 발생한 지점으로부터 계속된다.

단계(1001)에서, 만일 PC 리얼 타임 제어 기구가 실행중이 아니라고 판정되면 단계(1005)로 진행하고, 단계(1003)과 동일하게 과거에 계상된 정보에 의해 조건을 만족하는지 여부를 체크한다. 이 단계는 일정 시간내에서의 CPU점유율의 상한치 이상의 경우에 유효해지는 처리이다.

단계(1005)에서, 조건을 만족하고 있다고 판정된 경우는 사용권 감시 수단을 종료하고 PC-OS로 되돌아간다.

단계(1005)에서, 조건을 만족하지 않는, 즉 PC 리얼 타임 제어 기구는 동작 가능하다고 판정된 경우에는, 단계(1006)으로 진행하고, 제2 스케줄링 수단을 호출한다. 모든 실행 가능한 RT 프로세스의 처리가 종료하면, 제2 스케줄링 수단으로부터 제어가 단계(1007)로 전달되고, PC 리얼 타임 제어 기구의 동작 시간의 계상을 행한 후, PC-OS에 제어를 되돌린다.

다음에, PC-OS에 CPU 사용권을 강제적으로 반환하는 다른 처리예에 대하여, 도12에 따라 설명한다.

도면에서, 단계(1001-1007)은, 도10의 흐름과 동일한 처리이다.

단계(1003)에서, 조건을 만족하고 있다고 판단되면 단계(1201)로 진행하고, 현재 실행 가능 상태에 있는 RT 프로세스를 모두 실행 가능 상태로부터 벗어나게 하여, 사용권 감시 수단의 처리를 종료한다.

또한, 단계(1006)에서 조건을 만족하지 않는다고 판단된 경우에는, 단계(1203)에서 단계(1201)에서 유지한 RT 프로세스를 모두 실행 가능 상태로 되돌린 후, 단계(1005)에서 제2 스케줄링 수단을 호출하도록 한다.

[제3 실시예]

다음에 본 발명의 제3 실시예에 대하여, 도13 및 도14에 기초하여 설명한다.

도13에서, 도면 번호(1301a, 1301b)는 PC 리얼 타임 제어 기구간에 걸쳐 RT 프로세스가 데이터 교환·동기 처리를 행하는 경우의 배타 제어를 실행하는 경합 제어 수단이다.

그 외의 구성 요소는 도1에 도시한 바와 동일하다.

다음에 동작에 대하여 도14의 흐름도에 따라 설명한다.

경합 제어 수단은, 본 실시예에서는 RT 프로세스간 통신, 동기 등의 기능을 다른 PC 리얼 타임 제어 기구에 제공하기 위한 액세스 포인트이다. 프로세스간 통신, 동기 등 어떠한 기능을 PC 리얼 타임 제어 기구로서 제공할지는 자유이며, 본 발명과는 직접적인 관계가 없기 때문에, 여기서는 설명을 생략한다.

우선, 단계(1401)에서는 경합 제어 수단의 액세스 포인트에 도달된 처리 의뢰가, 자신의 PC 리얼 타임 제어 기구로부터 다른 PC 리얼 타임 제어 기구로의 처리 의뢰인지, 다른 PC 리얼 타임 제어 기구로부터의 처리 의뢰인지를 체크한다. 이 체크는 예를 들면, 처리 의뢰의 데이터 중에 이들을 구별 가능한 코드를 넣거나, PC 리얼 타임 제어 기구에 특정 번호를 붙이고, 이 번호를 처리 의뢰 데이터 중의 특정 위치에 매립하도록 하여도 좋다.

단계(1401)에서, 만일 다른 PC 리얼 타임 제어 기구에 처리를 의뢰하는 것이 없으면, 단계(1402)로 진행하고, 다른 PC 리얼 타임 제어 기구의 경합 제어 수단 「도13에서의 도면 부호(1301a)」의 액세스 포인트에 처리 의뢰 데이터를 송출한다.

단계(1401)에서, 만일 다른 PC 리얼 타임 제어 기구로부터의 처리 의뢰인 경우에는 단계(1403)로 진행하고, 의뢰된 처리를 실행하는 루틴을 호출한다.

[발명의 효과]

이상과 같이 본 발명에 의하면, 1개의 CPU상에서 동작하는 오퍼레이팅 시스템을 베이스로 하여, 어플리케이션 프로세스와 리얼 타임 프로세스를 동작시키도록 하였기 때문에, 버전업에도 용이하게 따를 수 있고, 하드웨어 비용을 억제한 프로세스 공존 시스템을 실현할 수 있다.

또한, 본 발명에 의하면, 리얼 타임 프로세스간 또는 리얼 타임 프로세스와 어플리케이션 프로세스간에서의 데이터 교환을 행하도록 하였기 때문에, 리얼 타임 프로세스와 어플리케이션 프로세스에서의 협조 동작을 실현할 수 있다.

또한, 본 발명에 의하면, 리얼 타임 프로세스를 어플리케이션 프로세스로 하여 개발한 후, 리얼 타임 프로세스로서 로딩하여 실행하도록 하였기 때문에, 실현해야 할 기능을 리얼 타임 프로세스로서 용이하게 프로그래밍할 수 있다.

또한, 본 발명에 의하면, 오퍼레이팅 시스템이 본래 입출력 장치의 디바이스 드라이버용으로 제공되어 있는 서비스를 리얼 타임 프로세스에 대해서도 제공하도록 하였기 때문에, 리얼 타임 프로세스가 어플리케이션 프로세스로서 개발된 것이었어도 안전하게 동작시킬 수 있다.

또한, 본 발명에 의하면, 리얼 타임 프로세스가 CPU 리소스를 점유하는 시간을 감시하고, 일정 시간 이상 점유하지 않도록 하였기 때문에, 어플리케이션 프로세스의 이용자에 대한 응답 성능을 손상하는 일 없이, 리얼 타임 프로세스를 실행할 수 있다.

또한, 본 발명에 의하면, 복수의 리얼 타임 제어 기구간에서의 액세스 경합 제어 기구를 실현하였기 때

문에, 1개의 오퍼레이팅 시스템 아래에 복수의 특성을 가진 리얼 타임 프로세스를 동시에 동작시킬 수 있다.

(57) 청구의 범위

청구항 1

어플리케이션 프로세스에 대하여 CPU 리소스의 할당 및 스케줄링을 행하는 제1 스케줄링 수단, 및 디바이스 장치에 대하여 입출력 처리를 행하는 디바이스 드라이버를 조립하는 드라이버 조립 기구를 구비한 오퍼레이팅 시스템에 있어서, 디바이스 장치로부터의 인터럽트에 대하여 리얼 타임 응답성이 요구되는 처리를 상기 오퍼레이팅 시스템으로부터 디바이스 드라이버로서 액세스되는 리얼 타임 제어 기구로서 구성하며, 상기 리얼 타임 제어 기구는 상기 디바이스 장치에 대한 처리 대응에 구성된 리얼 타임 프로세스, 및 상기 리얼 타임 프로세스에 대하여 상기 CPU 리소스의 할당 및 스케줄링을 행하는 제2 스케줄링 수단을 구비하는 것을 특징으로 하는 리얼타임 제어 시스템

청구항 2

제1항에 있어서, 상기 리얼 타임 제어 기구는 상기 제2 스케줄링 수단이 CPU 리소스를 오퍼레이팅 시스템으로부터 점유하고 있는 시간을 계측·유지하고, 상기 계측 결과에 기초하여 CPU 리소스를 강제적으로 오퍼레이팅 시스템으로 되돌리는 사용권 감시 수단을 구비하도록 한 것을 특징으로 하는 리얼 타임 제어 시스템.

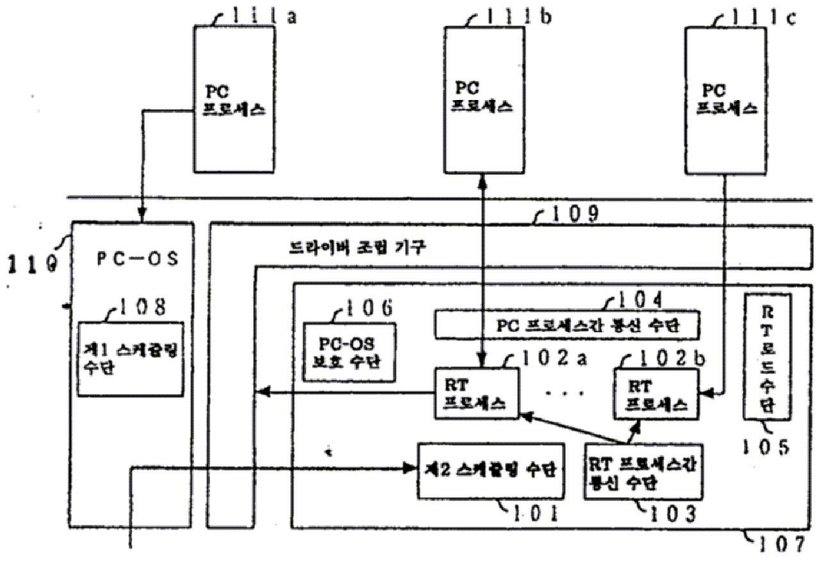
청구항 3

제1항 또는 제2항에 있어서, 상기 리얼 타임 제어 기구가 복수인 경우, 상기 복수의 리얼 타임 제어기구간의 주종 관계 또는 우선 순위 관계를 조정하는 경합 제어 수단을 구비하도록 한 것을 특징으로 하는 리얼 타임 제어 시스템.

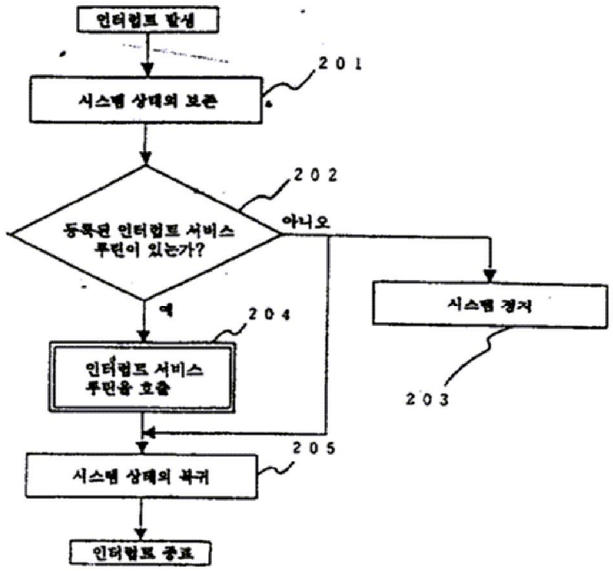
※ 참고사항 : 최초출원 내용에 의하여 공개하는 것임.

도면

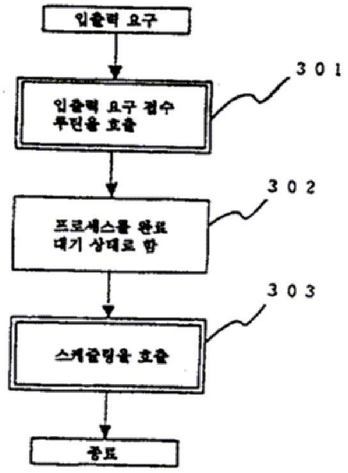
도면1



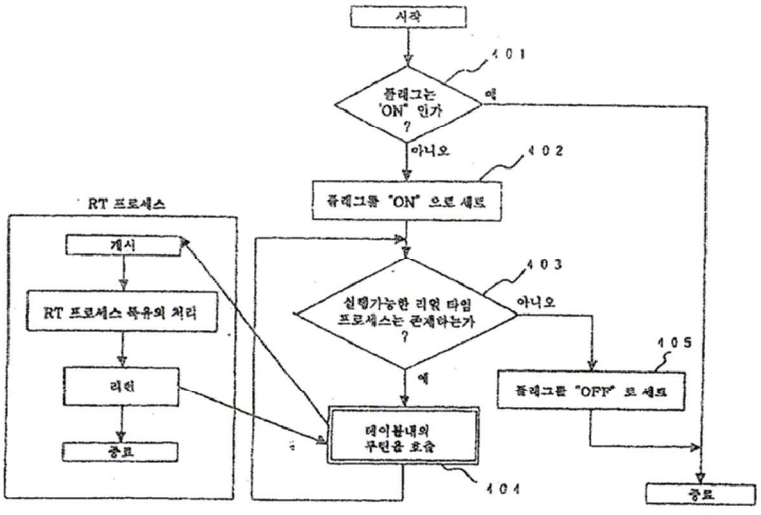
도면2


인터럽트 발생
201
시스템 상태의 보존
202
등록된 인터럽트 서비스 루틴이 있는가?
아니오
예
시스템 정지
203
204
인터럽트 서비스 루틴을 호출
205
시스템 상태의 복귀
인터럽트 종료

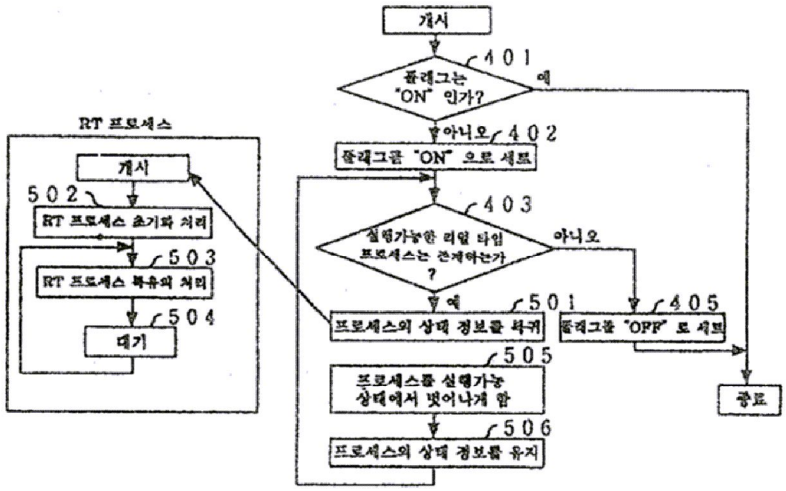
도면3


입출력 요구
301
입출력 요구 접수 루틴을 호출
302
프로세스를 완료 대기 상태로 함
303
스케줄링을 호출
종료

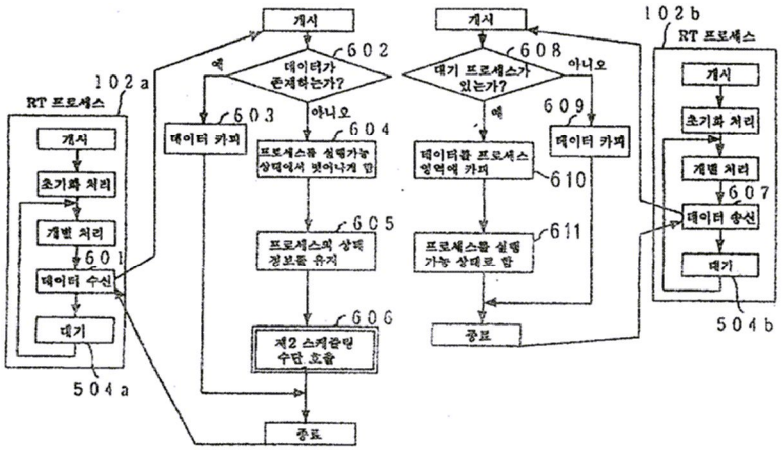
도면4


시작
401
플래그는 'ON' 인가 ?
예
아니오
402
플래그를 "ON" 으로 세트
RT 프로세스
게시
RT 프로세스 특유의 처리
리턴
종료
403
실행가능한 리얼 타임 프로세스는 존재하는가 ?
아니오
예
405
플래그를 "OFF" 로 세트
데이블내의 루틴을 호출
404
종료

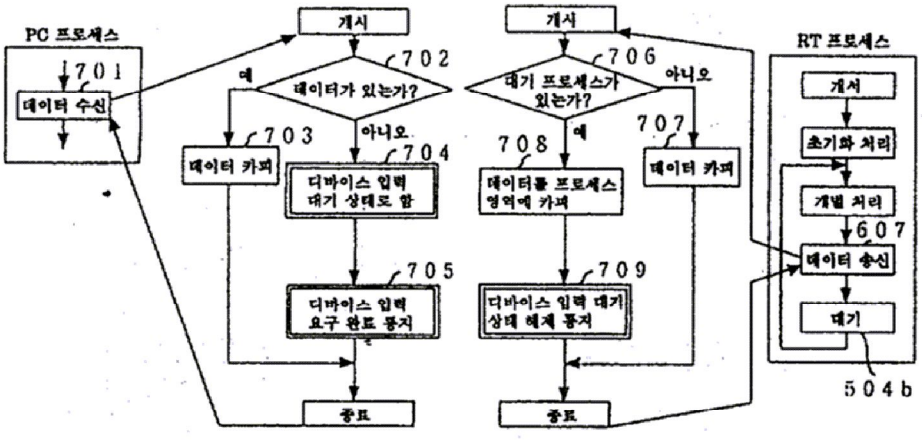
도면5


개시
401
플래그는 "ON" 인가?
예
아니오
402
플래그를 "ON" 으로 세트
403
실행가능한 리얼 타임 프로세스는 존재하는가?
아니오
예
501
405
프로세스의 상태 정보를 복귀
플래그를 "OFF" 로 세트
505
프로세스를 실행가능 상태에서 벗어나게 함
506
프로세스의 상태 정보를 유지
종료
RT 프로세스
개시
502
RT 프로세스 초기화 처리
503
RT 프로세스 특유의 처리
504
대기

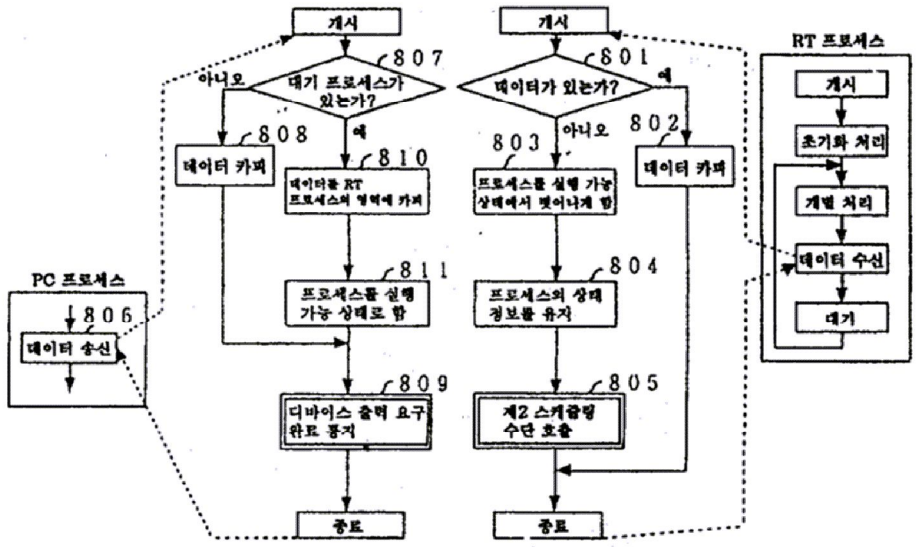
도면6


102a
RT 프로세스
개시
초기화 처리
개별 처리
601
데이터 수신
대기
504a
602
데이터가 존재하는가?
예
아니오
603
데이터 카피
604
프로세스를 실행가능 상태에서 벗어나게 함
605
프로세스의 상태 정보를 유지
606
제2 스케줄링 수단 호출
종료
608
대기 프로세스가 있는가?
아니오
예
609
데이터 카피
610
데이터를 프로세스 영역에 카피
611
프로세스를 실행 가능 상태로 함
종료
102b
RT 프로세스
개시
초기화 처리
개별 처리
607
데이터 송신
대기
504b

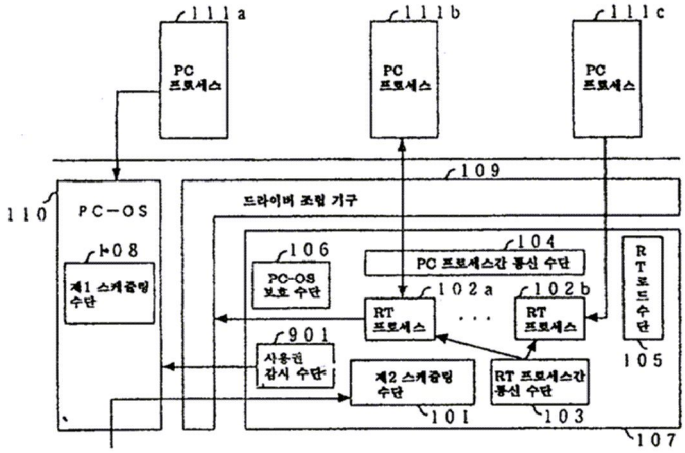
도면7


PC 프로세스
701
데이터 수신
개시
702
데이터가 있는가?
예
아니오
703
데이터 카피
704
디바이스 입력 대기 상태로 함
705
디바이스 입력 요구 완료 통지
종료
706
대기 프로세스가 있는가?
아니오
예
707
데이터 카피
708
데이터를 프로세스 영역에 카피
709
디바이스 입력 대기 상태 해제 통지
종료
RT 프로세스
개시
초기화 처리
개별 처리
607
데이터 송신
대기
504b

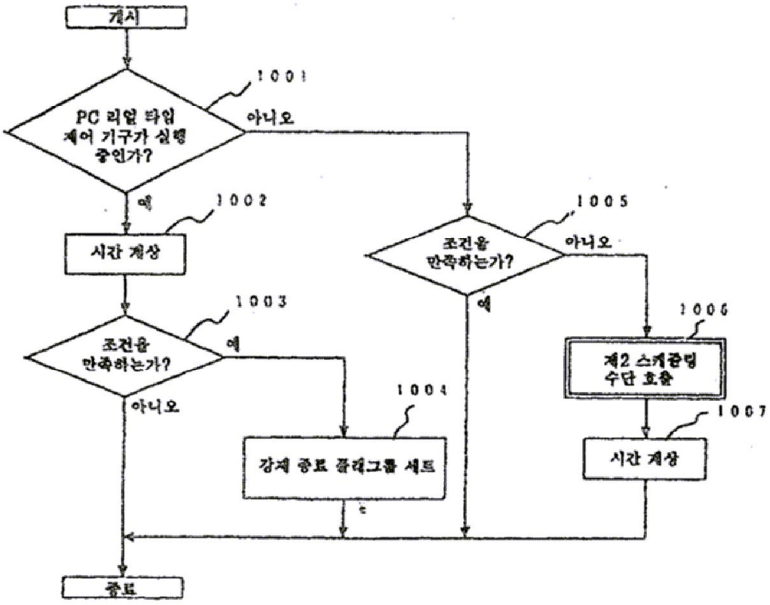
도면8


PC 프로세스
806
데이터 송신
게시
807
대기 프로세스가 있는가?
아니오
예
808
데이터 카피
810
데이터를 RT 프로세스의 영역에 카피
811
프로세스를 실행 가능 상태로 함
809
디바이스 출력 요구 완료 통지
종료
게시
801
데이터가 있는가?
예
아니오
803
802
데이터 카피
프로세스를 실행 가능 상태에서 벗어나게 함
804
프로세스의 상태 정보를 유지
805
제2 스케줄링 수단 호출
종료
RT 프로세스
게시
초기화 처리
계열 처리
데이터 수신
대기

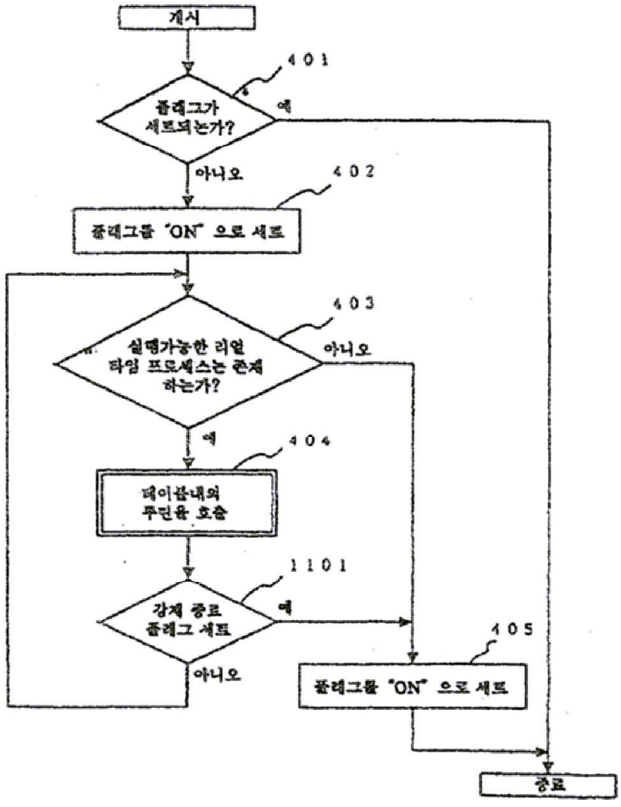
도면9


111a
PC 프로세스
111b
PC 프로세스
111c
PC 프로세스
109
110
PC-OS
드라이버 조립 기구
108
제1 스케줄링 수단
106
PC-OS 보호 수단
104
PC 프로세스간 통신 수단
102a
RT 프로세스
102b
RT 프로세스
RT 로드 수단
105
901
사용권 감시 수단
제2 스케줄링 수단
101
RT 프로세스간 통신 수단
103
107

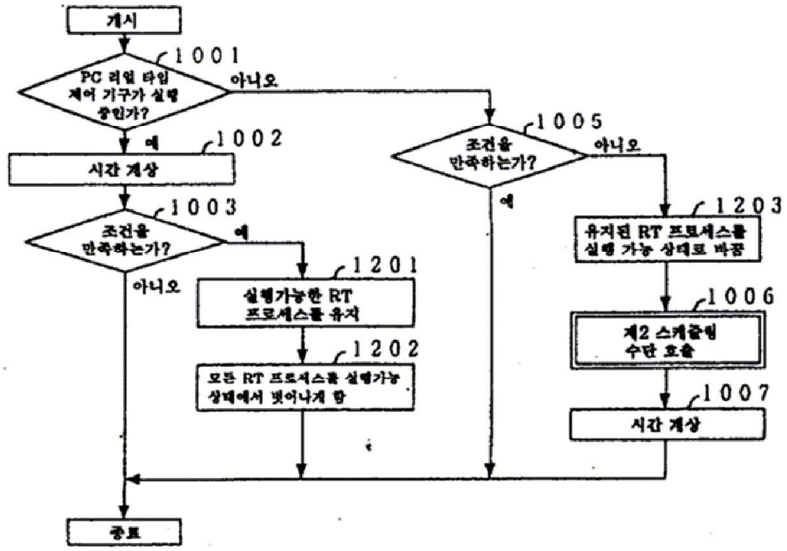
도면10


개시
1001
PC 리얼 타임 제어 기구가 실행 중인가?
아니오
예
1002
시간 계상
1003
조건을 만족하는가?
예
아니오
1004
강제 종료 플래그를 세트
1005
조건을 만족하는가?
아니오
예
1006
제2 스케줄링 수단 호출
1007
시간 계상
종료

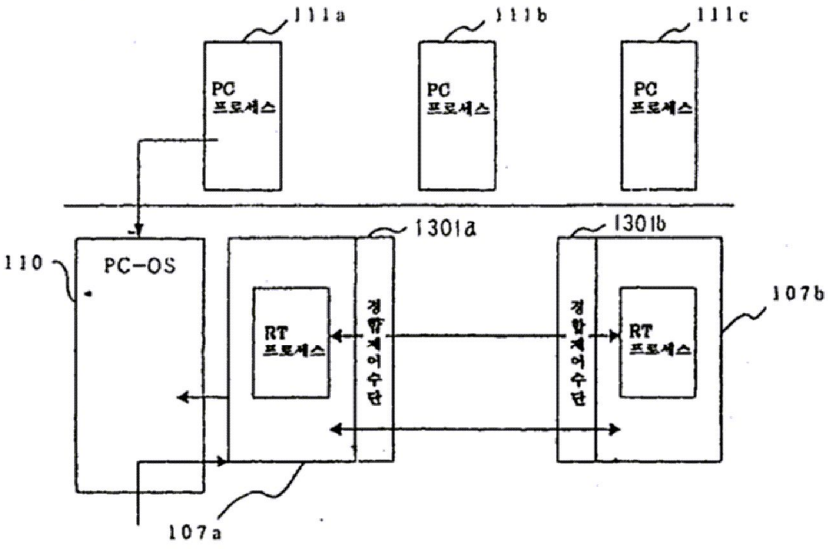
도면11


개시
401
플래그가 세트되는가?
예
아니오
402
플래그를 "ON" 으로 세트
403
실행가능한 리얼 타임 프로세스는 존재하는가?
아니오
예
404
데이플내의 루틴을 호출
1101
강제 종료 플래그 세트
예
아니오
405
플래그를 "ON" 으로 세트
종료

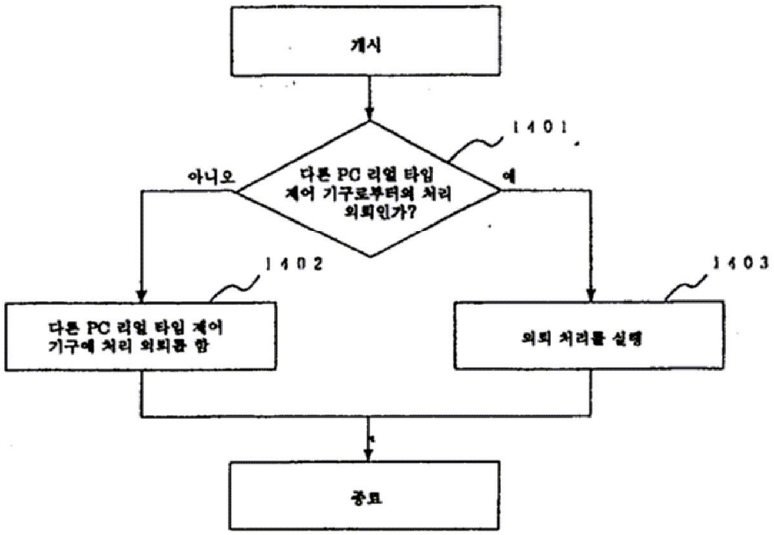
도면12


개시
1001
PC 리얼 타임 제어 기구가 실행 중인가?
아니오
예
1002
시간 계상
1003
조건을 만족하는가?
예
아니오
1201
실행가능한 RT 프로세스를 유지
1202
모든 RT 프로세스를 실행가능 상태에서 벗어나게 함
종료
1005
조건을 만족하는가?
아니오
예
1203
유지된 RT 프로세스를 실행 가능 상태로 바꿈
1006
제2 스케줄링 수단 호출
1007
시간 계상

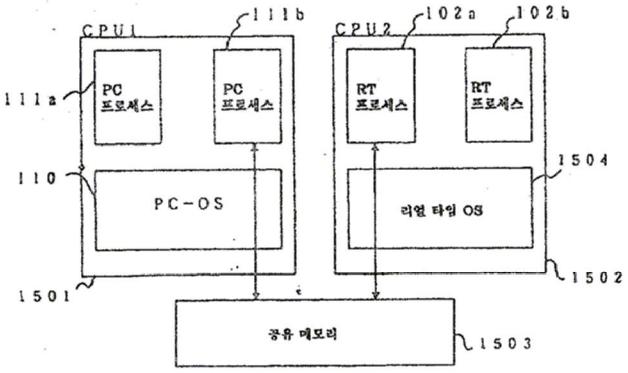
도면13


111a
111b
111c
PC 프로세스
PC 프로세스
PC 프로세스
1301a
1301b
110
PC-OS
RT 프로세스
정합제어수단
정합제어수단
RT 프로세스
107b
107a

도면14



도면15



도면16