(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. [6]
G06F 9/00

(45) 공고일자 2001년10월19일
(11) 등록번호 10-0306456
(24) 등록일자 2001년08월09일

(21) 출원번호 10-1998-0026409
(22) 출원일자 1998년07월01일

(65) 공개번호 특1999-0013514
(43) 공개일자 1999년02월25일

(30) 우선권주장 97-191840 1997년07월02일 일본(JP)

(73) 특허권자
가부시끼가이샤 히다치 세이사꾸쇼
가나이 쓰도무
일본국 도쿄도 지요다구 간다 스루가다이 4-6

(72) 발명자
세끼구찌 도모끼
일본 가나가와껭 요꼬하마시 아오바꾸 우쯔꾸시가오까니시 2-40-1히다찌 우쯔꾸시가오까료 에스115
아라이 도시아끼
일본 도꾜도 마찌다시 나루세가오까 3-31-5

(74) 대리인
장수길
주성민

심사관 : 전현진

(54) 오퍼레이팅시스템의재기동방법

요약

복수의 로드 모듈로 구성되는 오퍼레이팅 시스템의 1대의 계산기에서의 재기동 방법이며, 우선 이 오퍼레이팅 시스템의 재기동중에 동작되는 1개의 로드 모듈은 메모리에 보유해 둔다. 또한, 이 1개의 로드 모듈이 처리하는 인터럽트는 접수 가능한 상태로 해 둔다. 그리고, 이 1개 이외의 로드 모듈을 상기 계산기의 메모리에 로드한다. 이에 따라, 오퍼레이팅 시스템의 재기동중에도 이 1개의 로드 모듈에 의한 인터럽트의 처리가 가능해진다.

대표도
도 1

명세서

도면의 간단한 설명

도 1은 본 발명의 실시 형태가 적용되는 계산기의 구성도.

도 2는 로드 모듈 및 빈 메모리 공간을 관리하는 데이타 구조도.

도 3은 하드웨어 점유 자원을 관리하는 데이타 구조도.

도 4는 페이지 테이블 및 인터럽트 핸들러 테이블의 데이타 구조도.

도 5는 무정지 모듈을 관리하는 데이타 구조도.

도 6은 모듈 구성 파일.

도 7은 모듈을 저장하고 있는 로드 모듈 파일.

도 8은 소프트웨어 장애 검출 시의 처리의 플로우차트.

도 9는 커넬 재기동 처리의 제1 플로우차트.

도 10은 커넬 재기동 처리의 제2 플로우차트.

도 11은 무정지 모듈의 초기화 순서의 플로우차트.

도 12는 재기동 로더의 처리의 플로우차트.

도 13은 본 발명의 제2 실시 형태의 계산기의 기동 순서의 처리의 플로우차트.

도 14는 본 발명의 제2 실시 형태의 커넬 재기동 처리의 플로우차트.

< 도면의 주요 부분에 대한 부호의 설명 >

100 : 계산기

101 : 프로세서

102 : 주기억

103 : 판독 전용 메모리

104 : 인터럽트 컨트롤러

105 : 자기 디스크

106, 107 : 외부 디바이스

108 : 버스

109 : 인터럽트 신호 버스

111 : 커넬

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 오퍼레이팅 시스템의 재기동 방법에 관한 것으로, 특히 오퍼레이팅 시스템이 소프트웨어 장애에 의해 정지할 때의 오퍼레이팅 시스템의 재기동 방법에 관한 것이다.

어느 종류의 오퍼레이팅 시스템에서는, 계산기에 접속하고 있는 하드웨어를 제어하는 디바이스 드라이버 모듈을 커넬로부터 분리한다. 그리고, 커넬이 필요한 디바이스 드라이버를 자기 디스크 장치 등의 이차 기억 장치로부터 로드하여 이용한다. 상기한 기능을 갖는 오퍼레이팅 시스템에 있어서는 클럭 인터럽트를 오퍼레이팅 시스템으로부터 끼워 넣어 관리하는 실시간 처리 드라이버를 이용하여 시간 제약이 엄격한 시스템의 제어를 계산기에 실시시키는 방식이 있다. 이 방식으로는 원래 오퍼레이팅 시스템이 받아야 할 클럭 인터럽트를 실시간 처리 드라이버가 끼워 넣어 오퍼레이팅 시스템의 처리보다 우선하여 실시간 처리를 실시한다. 그 후 오퍼레이팅 시스템에 제어를 복귀한다.

이들 실현예로서는 'The RTX Real-Time Subsystem for Windows NT'(USENIX Windows NT Workshop – August 11-13, 1997, pp33 – 37)에 진술되고 있다. 이 문헌에서는 OS의 일부 모듈의 변경과 특수한 디바이스 드라이버에 의해 실시간 제어의 대상이 되는 디바이스의 인터럽트를 OS로부터 끼워 넣어 OS와는 관계없는 프로그램이 인터럽트 처리를 실시한다. 이에 따라, OS의 처리와는 관계없이 인터럽트를 처리할 수 있으며 실시간성을 높이고 있다.

실시간 처리에서는 신뢰성의 확보도 큰 문제가 된다. 상기한 방식에서는 실시간 처리 모듈을 오퍼레이팅 시스템의 커넬에 의존하지 않도록 한다. 이에 따라, 오퍼레이팅 시스템이 소프트웨어 장애에 의해 정지하여 버려도 실시간 처리를 계속할 수 있다. 또한, 오퍼레이팅 시스템이 장애에 의해 정지할 때에 실시간 처리 모듈에 그 취지를 통지한다. 이에 따라, 실시간 처리 모듈측에서 오퍼레이팅 시스템의 정지에 대처하는 처리를 실시할 수 있도록 하고 있다. 상기한 실현예에서도 실시간 제어 대상인 디바이스의 인터럽트 처리는 OS나 장애에 의해 정지해도 계속되도록 제어되고 있다.

발명이 이루고자 하는 기술적 과제

그러나, 종래의 방식으로는 소프트웨어 장애에 의해 정지하게 되는 오퍼레이팅 시스템을 재기동할 때 실시간 처리 드라이버도 정지하여 버린다. 즉, 오퍼레이팅 시스템의 재기동 처리와 실시간 처리를 동시에 실시할 수 없다고 하는 문제점이 있다. 이것은 오퍼레이팅 시스템의 재기동 시에 프로세서를 리셋트하게 되기 때문에 실시간 모듈이 동작하기 위한 가상 기억 내용, 인터럽트 처리용 정보가 소실되어 버리기 때문이다. 이것은 매우 짧은 주기에서 정지하는 일 없이 제어해야 하는 하드웨어가 있는 경우에는 오퍼레이팅 시스템의 재기동 시에 그 제어가 중단되어 버리기 때문에 문제가 된다. 종래 기술에서는 클럭 인터럽트에 한정되지 않고 오퍼레이팅 시스템의 재기동중에 오퍼레이팅 시스템은 하드웨어가 발생하는 외부 인터럽트를 접수할 수 없다.

예를 들면, 복수의 계산기로 이루어지는 클러스터(cluster) 구성의 계산기 시스템으로는 다른 계산기가 가동하고 있는지의 여부를 일정 간격으로 조회하여 일정 시간 대답이 없으면 그 계산기가 실행을 정지하고 있다고 판단하여 시스템 구성을 변경하는 처리를 실시한다. 이 경우에, 계산기가 정지하고 있다는 판단을 내리기 위해서는 일정한 대기 시간이 필요하게 된다. 오퍼레이팅 시스템의 재기동중에 인터럽트의 처리를 할 수 없다면 계산기가 정지하고 있다고 판단하는 일정한 대기 시간을 긴 시간으로 설정할 필요가 있다. 이 때문에, 시스템 재구성을 개시하기까지의 시간이 길어져서 문제가 된다. 이 조회에 반응하는 외부 인터럽트를 커넬 재기동 처리 중에도 접수하여 대답을 되돌려 줄 수 있으면, 전술

한 일정 시간을 짧게 할 수 있다. 따라서, 재구성을 개시하기까지의 시간을 단축할 수 있다.

또한, 종래의 재기동 방법으로는 메모리 체크, 하드웨어 구성 인식 등의 처리가 실행되기 때문에, 오퍼레이팅 시스템의 기동까지의 시간이 길어진다고 하는 문제가 있다.

본 발명의 목적은 오퍼레이팅 시스템의 재기동 시에서도 오퍼레이팅 시스템과는 독립해서 처리를 실시할 수 있는 외부 인터럽트 처리를 실행할 수 있도록 하는 것에 있다.

상기 목적을 달성하기 위해서 본 발명은 오퍼레이팅 시스템이 소프트웨어 장애를 검출하여 재기동을 하는 경우에, 재기동중에 동작시키는 로드 모듈은 메모리에 보유하고, 이 로드 모듈이 처리하는 외부 인터럽트는 접수 가능한 상태로서 상기 외부 인터럽트 이외는 인터럽트 금지 상태에서 상기 로드 모듈 이외의 로드 모듈을 로드한다.

발명의 구성 및 작용

이하에, 도면을 이용해서 본 발명의 제1 및 제2 실시 형태를 설명한다.

〈제1 실시 형태〉

도 1은 본 발명의 제1 실시 형태에서 사용되는 계산기의 구성도이다. 계산기(100)는 프로세서(101), 주기억(102), 판독 전용 메모리(103), 자기 디스크(105), 외부 디바이스(106, 107) 인터럽트 컨트롤러(104), 각 구성 요소를 접속하는 버스(108), 인터럽트 신호 버스(109)로 이루어진다.

판독 전용 메모리(103)에는 계산기의 초기화 처리를 실행하는 프로그램이 저장되어 있으며 프로세서(101)가 리셋트 상태가 될 때에 제어가 이동되는 어드레스에 초기화 프로그램이 배치되어 있다. 판독 전용 메모리(103)에 저장되어 있는 프로그램은 하드웨어 구성을 나타내는 데이타를 주기억 내에 구축한다. 또한, 자기 디스크가 미리 정해진 영역에 있는 데이타를 주기억에 로드하고 그것을 프로그램으로 간주하여 제어를 전달한다. 이 예에서는 로더(114)가 자기 디스크(105)가 정해진 영역에 있으며 이것이 주기억(102)에 로드되어 실행된다.

처리를 계속한 로더(114)는 커넬(111)을 주기억에 로드하고, 프로세서의 가상 어드레스 변환 기구를 설정하여 프로세서를 가상 어드레스 모드로 이동시켜 커넬(111)을 실행한다. 이때, 로더는 하드웨어 구성 데이타를 커넬(111)로 전달한다. 커넬(111)은 하드웨어 구성 데이타를 참조하여 커넬(111)이 관리하는 하드웨어 디바이스의 초기화 처리를 실시하고 모듈 구성 파일(110)의 내용에 따라서 커넬(111)로부터 분리된 프로그램 모듈을 주기억(102)에 로드하여, 각각의 모듈의 초기화 처리를 실행한다. 예를 들면, 모듈 구성 파일(110)이 재기동 로더(113), 디바이스 드라이버(112)를 로드하도록 기록되어 있으면, 커넬(111)은 이들을 주기억에 로드하여 각각의 초기화 루틴을 실행한다. 도 1은 주기억에 커넬(111), 재기동 로더(113) 및 디바이스 드라이버가 로드되어 있는 것을 나타내고 있다. 마지막으로, 커넬(111)은 최초의 프로세스를 생성하여 계산기 기동 순서를 완료한다.

이러한 계산기 시스템으로 오퍼레이팅 시스템 커넬(111)이 소프트웨어 장애에 의해 정지하게 될 때에, 프로세서를 리셋트하여 상기한 순서로 오퍼레이팅 시스템을 재기동하게 되면, 재기동 사이의 가상 어드레스 모드는 해제되어 외부 디바이스(106, 107)로부터의 인터럽트를 접수할 수 없게 된다. 여기서는 외부 디바이스(106)가 발생하는 인터럽트는, 커넬(111)이 소프트웨어 장애에 의해 정지하고 재기동하고 있는 동안 처리해야만 하는 인터럽트로서, 디바이스 드라이버(112)에 의해 관리되는 것으로서 설명한다. 본 발명은 커넬(111)의 재기동 처리를 하고 있는 동안도 특정한 인터럽트를 계속 접수하여 인터럽트 처리를 실행할 수 있는 재기동 순서를 제공하는 것이다.

디바이스 드라이버(112)는 커넬(111)이 제공하는 서비스를 이용하지 않고서 외부 디바이스(106)의 인터럽트를 처리하도록 프로그램되어 있는 모듈이다. 여기서는 디바이스 드라이버(112)는 무정지 모듈로서 재기동 로더(113)의 무정지 모듈 관리 테이블(500)에 등록된다. 재기동 로더(113)는 커넬(111)이 소프트웨어 장애에 의해 정지하여 커넬(111)의 재기동을 실시할 때에 실행되는 모듈이며, 무정지 모듈로서 관리 테이블(500)에 등록되어 있는 모듈의 처리 환경을 유지한 채로 커넬(111)의 재기동 처리를 실행한다.

무정지 모듈의 처리 환경이란, 무정지 모듈이 배치되어 있는 가상 어드레스, 그에 대한 어드레스 변환 테이블, 모듈이 저장되어 있는 물리 메모리, 무정지 모듈이 처리해야 할 인터럽트의 인터럽트 핸들러의 설정을 나타내고 있다. 어드레스 변환테이블은 페이지 테이블(410)에 설정되며, 인터럽트 핸들러는 인터럽트 핸들러 테이블(430)에 설정되어 있다. 또한, 어드레스 공간 정보와 하드웨어 자원 정보는 자원 관리 데이타(115)에 저장되어 있다.

재기동 로더(113)는 무정지 모듈 관리 테이블(500)을 참조해서 무정지 모듈이 확보하고 있는 어드레스 영역, 인터럽트 핸들러의 설정을 유지한채로 커넬(111)의 재로드 및 실행을 행한다. 이에 따라, 무정지 모듈의 인터럽트 처리의 연속성을 유지할 수 있다.

도 2는 본 발명의 오퍼레이팅 시스템의 커넬(111)의 어드레스 공간을 표현하는 데이타 구조를 나타내고 있다.

200은 커넬(111)의 어드레스 공간에 로드되어 있는 모듈을 기록하는 로드 모듈 관리 테이블이다. 각 모듈은 파일로서 자기 디스크 등의 이차 기억 장치에 저장되어 있으며, 오퍼레이팅 시스템 로드 또는 커넬(111)에 의해 주기억에 로드된다. 로드 모듈 관리 테이블(200)은 주기억에 로드되어 있는 각 모듈의 모듈명(201), 모듈의 코드 영역 개시 가상 어드레스(202), 코드 영역의 사이즈(203), 데이타 영역 개시 가상 어드레스(204), 데이타 영역의 크기(205) 및 모듈 인터페이스 정보(206)를 포함하고 있다.

모듈 인터페이스 정보(206)는 로드되는 모듈이 커넬(111)에 제공하는 인터페이스 루틴의 개시 어드레스이다. 인터페이스 루틴(206)을 등록함으로써 커넬(111)은 로드된 모듈의 기능을 이용할 수 있게 된다.

도 2의 207로부터 211은 모듈 명칭이 디바이스 드라이버(1)인 모듈의 코드 개시 어드레스, 사이즈, 데이타 개시 어드레스, 사이즈 및 인터페이스 루틴 정보를 저장하고 있다. 이 도면의 로드 모듈 관리 테이블(200)에서는 커넬(111), 재기동 로더(113), 디바이스 드라이버(1) 및 디바이스 드라이버(2)가 커넬 공간에 로드되어 있는 것을 나타내고 있다.

220은 커넬(111)의 가상 어드레스 공간의 공간 영역을 표현하는 커넬 가상 공간 빈 영역 리스트를 보유하고 있다. 리스트(220)는 공간 영역을 표현하는 공간 블럭 구조체(230)를 가리키고 있다. 빈 블럭 구조체(230)는 빈 영역의 개시 가상 어드레스(232), 그 빈 영역의 사이즈(233) 및 다음 공간 영역을 표현하는 빈 블럭 구조체로의 포인터(231)를 포함하고 있다. 231은 다음 빈 블럭 구조체(240)의 어드레스를 저장하고 있다. 이 도면의 220에서는 각각 어드레스(232, 242)로부터 시작해서 크기가 233, 243인 2개의 빈 영역이 커넬(111)의 가상 어드레스 공간에 있는 것을 나타내고 있다.

250은 물리 메모리의 빈 영역을 표현하는 물리 메모리 빈 영역 리스트이다. 물리 메모리 빈 영역 리스트(250)도 커넬 가상 공간 빈 영역 리스트(220)와 마찬가지로 구성된다. 빈 블럭 구조체(260, 270)가 각각 물리 메모리의 빈 영역을 표현하고 있다. 이 도면의 리스트(250)도 220과 마찬가지로 2개의 빈 물리 메모리가 있는 것을 표현하고 있다.

다음에, 계산기에 접속하고 있는 외부 디바이스를 관리하는 데이타 구조에 대해 설명한다.

도 3은 본 발명의 계산기에 접속하고 있는 외부 디바이스가 점유하는 자원을 표현하는 데이타 구조를 나타내고 있다. 300은 외부 디바이스의 제어용 레지스터가 존재하는 어드레스 범위를 표현하는 디바이스 점유 어드레스 리스트이다. 디바이스 점유 어드레스 리스트(300)는 1개의 어드레스 범위를 표현하는 디바이스 점유 어드레스 구조체로 이루어지며 도 3에서는 310, 320, 330의 3개의 디바이스 점유 어드레스 구조체로 구성되어 있다. 디바이스 점유 어드레스 구조체(310)는 디바이스 제어 레지스터 개시 어드레스(312), 크기(313), 그 어드레스 범위에 의해 제어되는 디바이스를 관리하고 있는 모듈의 모듈 번호(314) 및 디바이스 점유 어드레스 리스트를 구성하기 위한 링크(311)를 포함하고 있다.

모듈 번호(314)는 도 2의 로드 모듈 관리 테이블(200)로의 인덱스가 된다. 예를 들면, 디바이스 점유 어드레스 구조체(310)가 커넬(111)이 관리하고 있는 외부 디바이스의 점유 자원을 표현하고 있다고 하면, 314의 모듈 번호에는 테이블(200)에서의 커넬(111)의 인덱스인 0이 저장된다. 디바이스 드라이버(1)가 관리하는 디바이스가 있으면 그 모듈 번호에는 테이블(200)에서의 디바이스 드라이버(1)의 인덱스인 2가 저장된다. 또, 로드 모듈 관리 테이블(200)로의 인덱스는 0으로부터 세는 것으로 한다.

340은 외부 디바이스가 점유하는 인터럽트 번호를 기록하는 디바이스 점유 인터럽트 테이블이다. 디바이스 점유 인터럽트 테이블(340)은 프로세서가 인식하는 인터럽트 번호가 어느 모듈이 관리하는 하드웨어 디바이스에 점유되어 있는지를 기록하고 있다. 예를 들면, 커넬(111)이 관리하는 클럭 인터럽트용 디바이스가 인터럽트 번호 0을 점유하고 있으면 디바이스 점유 인터럽트 테이블의 0번째의 엔트리(341)에 커넬(111)의 모듈 번호인 0이 저장된다.

다음에, 프로세서가 직접 이용하는 데이타 구조에 대해 설명한다. 도 4는 프로세서가 이용하는 가상 어드레스 변환 테이블 및 인터럽트 처리용 테이블의 데이타 구조를 나타내고 있다. 410은 프로세서의 가상 어드레스로부터 물리 어드레스로의 어드레스 변환을 규정하는 페이지 테이블이다. 페이지 테이블(410)의 엔트리는 프로세서가 규정하는 페이지마다 존재한다. 각 엔트리는 가상 어드레스 공간 내의 각각의 가상 페이지에 대해 그 엔트리가 유효한지를 나타내는 플래그(411), 그 페이지가 기록 가능한지를 나타낸 플래그(412), 그 엔트리의 가상 페이지에 대응하는 물리 페이지의 개시 어드레스(413)를 포함하고 있다.

프로세서의 레지스터(400)는 페이지 테이블(410)의 개시 어드레스를 저장하고 있다. 프로세서는 가상 어드레스 모드에서 동작하고 있을 때에 페이지 테이블 레지스터(400)를 참조해서 가상 어드레스로부터 실제의 메모리 억세스에 필요한 물리 어드레스를 생성한다. 430은 프로세서에 들어가는 인터럽트에 대해 인터럽트 번호마다 인터럽트 핸들러를 규정하는 인터럽트 핸들러 테이블이다. 인터럽트에는 요인마다 번호가 할당되고 있다. 인터럽트 컨트롤러(104)는 외부 디바이스로부터의 인터럽트 요구를 받아 인터럽트 번호로 변환하여 프로세서에 통지한다. 인터럽트 핸들러 테이블(430)은 각각의 인터럽트 번호마다 인터럽트 핸들러의 개시 어드레스를 저장하고 있다. 예를 들면 클럭 인터럽트가 0번의 인터럽트를 이용하고 있으면 인터럽트 핸들러 테이블의 0번째의 엔트리(431)에는 클럭 인터럽트 핸들러의 어드레스가 저장되어 있다.

프로세서의 레지스터(420)는 인터럽트 핸들러 테이블(430)의 개시 어드레스를 저장하고 있다. 프로세서는 인터럽트 핸들러 테이블 레지스터(420)를 참조해서 인터럽트를 검출할 때에 요인마다의 인터럽트 핸들러로 제어를 이동시킨다. 예를 들면, 프로세서가 클럭 인터럽트 즉, 0번의 인터럽트를 검출할 때, 프로세서는 인터럽트 핸들러 테이블 레지스터(420)가 가리키는 테이블(430)의 0번 엔트리(431)에 저장되어 있는 핸들러로 제어를 이동시킨다.

다음에, 재기동 로더(113)가 관리하는 데이타 구조에 대해 설명한다. 재기동 로더(113)는 오퍼레이팅 시스템의 커넬(111)이 소프트웨어 장해에 의해 정지되고, 오퍼레이팅 시스템을 재기동할 때에 오퍼레이팅 시스템 커넬(111)을 주기억에 로드하여 커넬(111)을 실행하는 모듈이다.

도 5는 재기동 로더(113)가 관리하는 데이타 구조를 나타내고 있다. 500은 무정지 모듈 관리 테이블이며, 재기동 로더(113)에 의해 커넬(111)의 재로드, 재기동중에도 하드웨어 디바이스로부터의 인터럽트를 접수하여 처리를 실행하는 모듈인 무정지 모듈의 정보를 관리하는 데이타 구조이다. 무정지 모듈 관리 테이블(500)은 무정지 모듈명(501), 무정지 모듈의 코드 영역 개시 어드레스(502), 코드 영역의 크기(503), 데이타 영역의 개시 어드레스(504), 데이타 영역의 크기(505), 무정지 모듈이 관리하는 외부 디바이스가 점유하는 자원의 정보(506) 및 무정지 모듈의 재초기화 루틴의 어드레스를 포함하고 있다.

이용 하드웨어 정보(506)에는 무정지 모듈이 관리하는 외부 디바이스의 제어용 점유 어드레스, 점유 인터럽트 번호가 기록되어 있다. 재초기화 루틴(507)에는 커넬(111)이 재기동할 때 실행되는 루틴의 어드레스를 저장한다. 재초기화 루틴이 무엇을 실행할지는 모듈이 관리하고 있는 하드웨어에 의존하지만, 적어도 로드 모듈 관리 테이블(200)로의 인터페이스 루틴(206)의 등록을 실행한다. 이에 따라, 커넬(111)은 다시 무정지 모듈이 제공하는 기능을 이용할 수 있게 된다.

도 5의 무정지 모듈 관리 테이블(500)에서는 재기동 로더(113) 및 디바이스 드라이버(1)가 무정지 모듈로서 등록되어 있다. 또한, 508 내지 513에는 디바이스 드라이버(1)의 점유 어드레스, 외부 디바이스 정보, 재초기화 루틴 어드레스가 저장되어 있다. 재기동 로더(113)는 무정지 모듈 관리 테이블(500)의 정보를 이용해서 오퍼레이팅 시스템 커넬(111)의 재기동 시의 커넬 공간 및 외부 디바이스 구성 정보를 구축하여 커넬(111)에 전달된다. 커넬(111)이 이들의 정보를 참조해서 기동 순서를 결정함으로서 커넬 재기동중에도 중단하는 일 없이, 외부 디바이스로부터의 인터럽트 처리를 가능하게 한다. 무정지 모듈 관리 테이블(500)은 재기동 로더(113)의 데이타 영역에 놓여진 데이타 구조에서 재기동 로더 모듈에 의해 관리된다. 재기동 로더 모듈은 모듈의 초기화 시에 재기동 로더(113)의 데이타 영역을 기록 금지로 하고, 소프트웨어 장해에 의한 무정지 모듈 관리 테이블(500)의 파괴를 막는다. 데이타 영역을 기록 금지로 하기 위해서는 데이타 영역을 포함하는 가상 페이지에 대응하는 페이지 테이블(410)의 엔트리의 기록 가능 플래그를 리셋트하는 것으로 실현된다.

다음에, 커넬(111)이 기동시에 판독하는 로드 모듈을 정의하는 파일의 형식에 대해 설명한다. 도 6은 로드 모듈을 정의하는 데이타 구조를 나타내고 있다. 110은 모듈 구성 파일의 내용을 나타내고 있다. 모듈 구성 파일(110)의 각 엔트리는 커넬(111)이 로드하는 모듈의 명칭(601), 모듈이 저장되어 있는 파일명(602), 및 그 모듈이 무정지 모듈 인지를 나타낸 플래그(603)를 포함하고 있다. 도 6의 예에서는 커넬(111)은 재기동 로더(113), 디바이스 드라이버(1), 디바이스 드라이버(2)의 순서로 모듈을 주기억에 판독한다. 디바이스 드라이버(1)가 저장되어 있는 파일의 이름은 driver 1이고, 디바이스 드라이버(1)는 무정지 모듈인 것을 표현하고 있다. 모듈 구성 파일(110)은 오퍼레이팅 시스템이 미리 정한 이름의 파일에 저장되며 커넬(111)은 용이하게 이 파일을 발견할 수 있다.

도 7은 로드 모듈이 저장되는 파일의 형식을 나타내고 있다. 로드 모듈을 저장하고 있는 파일(700)은 모듈의 실행 코드가 저장되어 있는 장소를 나타낸 코드 개시 오프셋(701), 실행 코드의 사이즈(702), 데이타가 저장되어 있는 장소를 나타낸 데이타 개시 오프셋(703), 데이타 영역의 사이즈(704), 모듈의 초기화 루틴의 실행 코드가 저장되어 있는 초기화 루틴 오프셋(705), 모듈의 재배치 정보 개시 오프셋(706), 재배치 정보의 사이즈(707), 실행 코드(708) 및 데이타(709)를 보유하고 있다.

모듈의 재배치 정보(710)는 모듈을 주기억에 로드할 때에 이용하는 데이타로, 모듈의 코드 및 데이타 영역이 로드된 어드레스에 따라서 모듈의 코드를 변경하기 위한 데이타이다. 이에 따라, 로드하는 모듈의 구성이 변해서 모듈이 로드되는 어드레스가 변해도 좋다.

다음에, 본 발명의 오퍼레이팅 시스템 재기동 순서에 대해 설명한다. 도 8은 오퍼레이팅 시스템을 재기동할 때에 실행되는 재기동 로더(113)의 처리 순서를 나타낸 플로우차트이다. 우선, 스텝 801에서 커넬 공간에 재기동 로더(113)가 로드되어 있을지 검사한다. 재기동 로더(113)가 로드되어 있지 않거나 혹은 로드되어 있는지의 여부를 판별할 수 없는 경우는 스텝 802로 진행한다. 스텝 802에서는 프로세서를 리셋트하여 계산기의 재기동을 실행한다.

많은 계산기에서는 프로세서를 리셋트 상태로 하면 가상 어드레스 변환이 무효가 되며, 인터럽트 핸들러의 설정도 무효가 되며, 프로세서가 규정하는 물리 어드레스로 제어를 이동시킨다. 통상, 이 물리 어드레스에는 계산기의 기동 순서를 기억한 판독 전용 메모리(103)가 맵핑되어 있다. 이 기동 순서는 계산기에 접속되어 있는 하드웨어 디바이스를 리셋트한다. 이것은 오퍼레이팅 시스템의 커넬(111) 실행시에 외부 디바이스를 기지의 상태로 하기 때문이다. 이 기동 순서를 위해 특히 프로세서가 리셋트되어 버리기 때문에, 종래의 계산기에서는 오퍼레이팅 시스템 재기동 동안, 외부 디바이스로부터의 인터럽트 처리를 접수할 수 없게 된다.

한편, 스텝 801에서 재기동 로더(113)가 로드되어 있다고 판단된 경우는, 프로세서를 리셋트하는 일 없이 스텝 803으로 진행한다. 스텝 803으로부터가 실제의 재기동 로더(113)가 실행하는 처리이다. 스텝 803에서는 로드 모듈 관리 테이블(200)에 등록되어 있는 모듈이 하드웨어 리셋트 루틴을 갖고 있을지 검사하고 등록되어 있으면 그 리셋트 루틴을 호출한다. 특별한 하드웨어 리셋트가 필요없는 디바이스의 경우는 리셋트 루틴은 등록하지 않아도 좋다.

다음 스텝 804에서는 무정지 모듈 관리 테이블(500)의 각 엔트리의 이용 하드웨어 정보(506)를 참조해서 무정지 모듈이 접수되는 것 이외의 인터럽트에 대해 이들 인터럽트의 인터럽트 핸들러를 무효로 한다. 구체적으로는 인터럽트 핸들러 테이블(430)의 엔트리를 인터럽트를 접수하는 것뿐인 인터럽트 핸들러의 어드레스로 설정한다. 다음 스텝 805에서는 이 후에 계속되는 무정지 모듈의 데이타 영역으로의 데이타 저장에 대비하여 무정지 모듈의 데이타 영역의 저장되어 있는 페이지를 기록 가능하게 설정한다. 구체적으로는 페이지 테이블(410)의 무정지 모듈의 데이타 영역에 대응하는 엔트리의 기록 가능 플래그(412)를 기록 가능 상태로 셋트한다.

스텝 806에서는 가상 메모리 빈 영역 리스트 및 물리 메모리 빈 영역 리스트를 작성한다. 스텝 806에서는 무정지 모듈 관리 테이블(500)을 참조해서 무정지 모듈, 재기동 로드(113) 및 커넬 스택(116)이 이용하고 있는 이외의 어드레스 영역을 빈 영역으로 하는 빈 영역 리스트를 재기동 로더(113)의 데이타 영역 내로 작성한다. 빈 영역 리스트는 오퍼레이팅 시스템이 관리하는 빈 영역 리스트(220, 250)와 동일하게 데이타 구조를 갖는다.

스텝 807에서는 디바이스 점유 어드레스 리스트를 작성한다. 스텝 807에서는 스텝 806과 동일하게, 무정지 모듈 관리 테이블(500)을 참조해서 무정지 모듈이 이용하고 있는 이외의 어드레스 영역을 빈 영역으로 하는 디바이스 점유 어드레스 리스트를 재기동 로더(113)의 데이타 영역 내에 작성한다. 여기서, 작성하는 디바이스 점유 어드레스 리스트는 오퍼레이팅 시스템이 관리하는 디바이스 점유 어드레스 리스트(300)와 거의 동일한 데이타 구조이지만 디바이스 점유 어드레스 구조체의 모듈 번호에 대해서는 모듈 번호가 아닌 모듈 명칭을 저장한다.

스텝 808에서는 디바이스 점유 인터럽트 테이블을 작성한다. 스텝 807과 동일하게 무정지 모듈이 이용하고 있는 이외의 인터럽트 번호를 미사용으로 하는 디바이스 점유 인터럽트 테이블을 재기동 로더(113)의 데이타 영역 내에 작성한다. 여기서도 스텝 807과 동일하게 디바이스 점유 인터럽트 테이블의 각 엔트리에는 모듈 번호가 아닌 모듈 명칭을 저장한다. 스텝 806 내지 808에서 작성한 데이타 구조는 커넬(111)이 재기동할 때에 무정지 모듈이 관리하고 있는 하드웨어 디바이스 자원을 오류로 다른 모듈에 할당하지 않도록 하기 위한 데이타 구조이다.

마지막으로 재기동 로더(113)는 주기억에 오퍼레이팅 시스템 커넬(111)을 로드하고(스텝 809), 재기동 로더(113)의 재초기화 루틴의 어드레스를 파라메터에 더해 커넬(111)을 실행한다(스텝 810).

도 9는 본 발명의 실시예의 오퍼레이팅 시스템 커넬(111)의 초기화 처리 순서를 도시한 플로우차트이다.

우선, 커넬(111)은 기동 시에 재기동 로더(113)에 의한 기동인지 통상의 기동인지를 판별한다(스텝 901). 이것은 재기동 로더(113)에 의한 기동에서는 재기동 로더(113)의 재초기화 루틴의 어드레스를 파라메터로서 커넬(111)에 전달되므로 용이하게 판별할 수 있다.

재기동 로더(113)에 의한 기동이 아닌 경우는 통상의 순서에 의해 커넬(111)의 실행을 진행시킨다(스텝 902). 여기서의 통상의 순서란, 버스 및 인터럽트 컨트롤러의 초기화, 커넬 관리의 하드웨어 디바이스의 초기화, 모듈 구성 파일에 기록된 모듈의 주기억으로의 로드와 초기화, 초기 프로세스의 실행이다.

재기동 로더(113)에 의한 기동의 경우는 스텝 903으로 진행한다. 스텝 903에서는 재기동 로드 모듈용에 로드 모듈 관리 테이블(200)의 엔트리를 할당한다. 계속되는 스텝 904에서 파라메터로서 전달되는 재기동 로드 모듈의 재초기화 루틴의 어드레스를 기초로 재초기화 루틴을 실행한다. 재기동 로더(113)의 재초기화 루틴은 무정지 모듈의 재초기화 루틴과 마찬가지의 처리를 실시한다. 이 재초기화 루틴에서는 로드 모듈 관리 테이블(200)의 설정을 실시한다. 특히, 관리 테이블(200)의 모듈 인터페이스(206)가 설정되며, 재기동 루틴이 제공하는 인터페이스 루틴을 커넬(111)로부터 호출할 수 있게 된다.

계속해서 스텝 905에서 하드웨어 구성 정보를 복사한다. 또한, 스텝 906과 907에서 재기동 로더(113)가 커넬 기동 전에 구축한 빈 영역 리스트, 디바이스 점유 어드레스 리스트, 디바이스 점유 인터럽트 테이블을 커넬(111)의 데이타 공간에 복사한다. 이들의 처리는 재기동 로더(113)가 제공하는 인터페이스 루틴에 의해 실행된다.

통상 기동 시에는 커넬(111)이 로드되어 있는 이외의 메모리 영역을 빈 영역으로 하는 공간 영역 리스트, 비어있는 디바이스 점유 어드레스 리스트 및 모든 인터럽트의 엔트리가 미사용인 디바이스 인터럽트 테이블을 작성해서 이후의 처리를 실행한다.

그에 대해 재기동 로더(113)에 의한 커넬 재기동 시에는 재기동 로더(113)가 작성한 빈 영역 리스트, 디바이스 점유 어드레스 리스트 및 디바이스 점유 인터럽트 테이블을 이용한다. 이에 따라, 무정지 모듈의 데이타 영역의 초기화, 외부 디바이스의 초기화를 피하고, 무정지 모듈의 외부 디바이스에 관련하는 처리의 연속성을 유지하는 것이 가능해진다.

계속되는 스텝 908로부터 912는 커넬(111)이 관리하는 버스와 인터럽트 컨트롤러 이외의 모든 외부 디바이스에 대한 초기화 처리이다. 우선, 스텝 908에서는 커넬(111)이 관리하는 하드웨어 디바이스가 있는지를 판정하고, 없으면 스텝 1001로 진행한다. 스텝 909에서는 외부 디바이스가 이용하는 어드레스 영역 및 인터럽트 번호가 이미 이용되어 있는지를 검사한다. 혹시, 초기화 처리 중 외부 디바이스가 이용하는 어드레스, 인터럽트 번호가 이미 디바이스 점유 어드레스 리스트(300) 및 디바이스 점유 인터럽트 테이블(340)에, 등록되어 있는 경우는 스텝 910 및 911은 실행하지 않고서 스텝 912로 진행한다. 초기화 처리 중 외부 디바이스가 이용하는 어드레스 혹은 인터럽트 번호가 이미 디바이스 점유 어드레스 리스트(300) 및 디바이스 점유 인터럽트 테이블(340)에 등록되어 있으면 이것은 무정지 모듈이 이용하고 있는 디바이스인 것을 의미한다. 이에 따라, 무정지 모듈이 관리하고 있는 외부 디바이스의 초기화를 피하고, 무정지 모듈의 디바이스에 관련하는 처리의 연속성이 유지된다.

외부 디바이스가 사용하는 어드레스 영역 및 인터럽트 번호가 미사용이면 그 어드레스 영역 및 인터럽트 번호를 디바이스 점유 어드레스 리스트(300)와 디바이스 점유 인터럽트 테이블(340)에 커넬(111)의 모듈 번호와 함께 등록한다(스텝 910).

커넬(111)의 모듈 번호는 로드 모듈 관리 테이블(200)을 검색함으로써 얻어진다. 또한, 대상이 되어 있는 외부 디바이스의 초기화를 실시하고(스텝 911), 스텝912로 진행한다. 스텝 912에서는 처리 대상 디바이스를 다른 디바이스로 설정해서 스텝 908로 진행한다. 처리 대상 디바이스가 남아 있지 않으면, 스텝 908의 검사로부터 스텝 1001로 처리를 진행시킨다.

스텝 1001로부터의 처리의 플로우차트는 도 10에 도시한다. 도 10의 플로우차트는 주로 모듈의 로드 처리에 관한 것이다.

스텝 1001에서는 모듈 구성 파일(110)을 메모리에 기록한다. 계속되는 스텝 1002 내지 1011은 모듈 구성 파일에 등록되어 있는 각 모듈에 대해 실행한다. 스텝 1002와 1008에 의해 각 모듈마다 처리를 실행하도록 루프를 구성한다.

루프 내의 최초의 스텝 1003에서는 모듈 구성 파일(110)의 무정지 플래그(603)를 참조해서 처리 대상의 모듈이 무정지 모듈인지의 여부를 검사한다. 무정지 모듈이 아닌 경우는 스텝 1004로 진행한다.

스텝 1004로부터 시작되는 처리는 모듈의 주기억으로의 로드를 실시한다. 우선, 스텝 1004에서 필요한 메모리 영역을 획득한다. 이것은 빈 영역 리스트(220, 250)를 참조해서 모듈을 저장하는데 충분한 빈 영역을 검색하고, 찾아낸 영역을 빈 리스트로부터 제외하여 실시한다. 모듈을 저장하는데 필요한 영역은 모듈을 저장하고 있는 파일의 선두 부분(702, 704)을 참조하여 구한다. 메모리의 할당시에는, 페이지 테이블(410)의 설정도 행한다. 할당한 어드레스 영역에 대응하는 페이지 테이블(410) 각각의 엔트리에 대해 유효 플래그(411)를 셋트하고, 할당한 물리 페이지의 선두 어드레스를 413에 저장한다. 또한, 기록 가능 플래그(412)도 셋트한다. 코드 영역이 저장되어 있는 페이지의 기록 가능 플래그(412)는 코드 영역의 로드 후 리셋트한다.

계속되는 스텝 1005에서 모듈 구성 파일(110)의 파일명(602)에서 나타낸 파일을 스텝 1004에서 할당한 영역으로 로드한다. 로드 후, 파일에 저장되어 있는 재배치 정보(706, 707, 710)에 의해 코드 영역을 수정한다. 또한, 로드 모듈 관리 테이블(200)에 처리 중인 모듈용 엔트리를 할당하여, 그 엔트리의 어드레스 및 사이즈 정보를 설정한다(스텝 1006).

다음에, 모듈의 초기화 루틴을 실행한다(스텝 1007). 모듈의 초기화 루틴은 적어도 로드 모듈 관리 테이블(200)의 모듈 인터페이스(206)를 설정한다. 혹시, 그 외의 모듈이 필요로 하는 초기화 처리가 있으면 그것도 실행한다. 모듈 인터페이스(206)가 설정되면 커넬(111)은 모듈이 제공하는 처리 루틴의 어드레스를 알 수 있으며, 모듈이 제공하는 처리를 실행할 수 있게 된다.

계속되는 스텝 1008에서는 처리 대상 모듈을 다음 모듈 구성 파일의 엔트리로 설정해서 스텝 1002로 복귀한다. 다음에, 처리 대상 모듈이 무정지 모듈인 경우에 대해 설명한다. 스텝 1003에서 처리 대상 모듈이 무정지 모듈이라고 판정된 경우 스텝 1009로 진행한다. 스텝 1009에서는 로드 모듈 관리 테이블(200)의 갱신 처리를 한다. 우선, 로드 모듈 관리 테이블(200)에 처리 중인 모듈의 데이타를 저장하는 엔트리를 할당한다. 할당한 엔트리에 저장해야 할 로드 모듈 관리 테이블(200)의 202 내지 205의 어드레스 및 사이즈 정보는 재기동 로더(113)의 데이타 영역 내의 무정지 모듈 관리 테이블(500)의 502 내지 505에 저장되어 있으며, 이것을 할당한 엔트리에 복사하여 로드 모듈 관리 테이블(200)을 갱신한다.

다음에, 무정지 모듈의 재초기화 루틴을 실행한다(스텝 1010). 재초기화 루틴의 어드레스는 무정지 모듈 관리 테이블(500)의 재초기화 루틴(507)에 저장되어 있으며, 무정지 모듈 명칭으로부터 재초기화 루틴을 얻을 수 있다. 무정지 모듈의 재초기화 루틴은 적어도 로드 모듈 관리 테이블(200)의 인터페이스 루틴(206)의 설정을 실행한다. 필요하다면, 하드웨어 디바이스의 설정 등 그 밖의 처리를 행한다. 이 때에, 무정지 모듈은 새롭게 로드되지 않기 때문에 지금까지의 데이타 영역은 보존되며 무정지 모듈의 디바이스에 관련하는 처리의 연속성을 유지할 수 있다.

계속되는 스텝 1011에서는 디바이스 점유 어드레스 리스트(300) 및 디바이스 점유 인터럽트 테이블(340) 중, 처리 대상의 무정지 모듈이 관리하는 어드레스 영역 및 인터럽트 번호를 표현하고 있는 엔트리의 모듈 번호 란에 스텝 1009

에서 할당한 로드 모듈 관리 테이블(200)의 엔트리 번호를 저장한다. 무정지 모듈이 관리하는 디바이스 어드레스 영역과 인터럽트 번호는 무정지 모듈 관리 테이블(500)의 이용 하드웨어 정보(506)에서 얻어진다. 빈 영역 리스트(220, 250)에 대해서는 재기동. 로더(113)가 무정지 모듈이 있는 어드레스 영역을 빈 리스트로부터 제외하고 있기 때문에 특별한 처리를 할 필요는 없다. 계속해서 스텝 1008로 진행하고 다음 로드 모듈의 처리를 행한다.

스텝 1002에서 모든 모듈을 로드했다고 판정하면 스텝 1012로 진행한다. 스텝 1012는 초기 프로세스의 작성과 실행을 행하여 오퍼레이팅 시스템의 기동을 완료한다.

다음에, 무정지 모듈의 초기화 루틴에 대해 설명한다. 모듈의 초기화 루틴은 모든 모듈이 가져야만 하는 루틴으로서, 커넬(111)의 기동 시에 실행된다. 도 11은 본 발명의 실시 형태의, 무정지 모듈의 초기화 루틴의 처리를 나타낸 플로우차트이다. 여기서는 모듈 구성 파일(110)로 설정되어 있는 디바이스 드라이버(1)의 초기화 루틴의 처리를 설명한다.

우선, 디바이스 드라이버(1)의 커넬(111)로의 인터페이스가 되는 모듈 인터페이스를 로드 모듈 관리 테이블(200)의 디바이스 드라이버(1)용에 할당한 엔트리(211)에 저장한다(스텝 1101).

다음 스텝 1102에서는 디바이스 드라이버(1)용 디바이스 점유 어드레스 구조체를 할당하여 개시 어드레스, 사이즈, 모듈 번호를 설정하여 디바이스 점유 어드레스 리스트(300)에 추가한다. 또한, 디바이스 점유 인터럽트 테이블(340)의 디바이스 드라이버(1)가 관리하는 인터럽트 번호의 엔트리에 디바이스 드라이버(1)의 모듈 번호를 저장한다. 이에 따라, 디바이스 드라이버(1)가 이용하는 디바이스 어드레스 영역 및 인터럽트 번호를 디바이스 점유 어드레스 리스트(300) 및 디바이스 점유 인터럽트 테이블(340)에 등록한다.

계속해서, 모듈의 무정지 모듈 관리 테이블(500)로의 등록을 행한다(스텝 1103). 무정지 모듈 관리 테이블에 현재 처리 중인 모듈용 엔트리를 할당하여 어드레스, 사이즈 정보, 이용 하드웨어 정보 및 재초기화 루틴의 어드레스를 설정한다. 설정은 재기동 로더(113)가 제공하는 인터페이스 루틴에 의해 실행한다. 무정지 모듈 관리 테이블(500)을 갱신하는 경우 무정지 모듈 관리 테이블(500)이 있는 페이지는 기록 금지로 설정되어 있으므로, 갱신의 전후로 페이지 테이블(410)의 기록 가능 플래그를 조작한다. 그리고, 하드웨어 디바이스를 설정한다(스텝 1104).

다음에, 재기동 로더(113)가 제공하는 인터페이스 루틴에 대해 설명한다. 재기동 로더(113)는 그 자신이 무정지 모듈이며, 초기화 루틴, 재초기화 루틴, 하드웨어 구성 데이타 복사 루틴, 무정지 모듈 등록 루틴을 제공한다.

우선, 재기동 로더(113)의 초기화 루틴의 처리에 대해 설명한다. 도 12는 재기동 로더(113)의 처리 순서를 나타낸 플로우차트이다.

재기동 로더(113)는 무정지 모듈이며 재기동 로더(113)의 초기화 루틴은 계산기가 기동될 때에만 불린다. 우선, 재기동 로더(113)의 초기화 루틴으로는 이후의 재기동 처리에 대비하여 판독 전용 메모리(103)에 저장되어 있는 초기 프로그램이 작성한 하드웨어 구성 정보를 재기동 로더(113)의 데이타 영역에 복사한다(스텝 1201). 다음에, 무정지 모듈 관리 테이블(500)의 초기화(스텝 1202), 재기동 로더(113) 자신의 무정지 모듈 관리 테이블로의 등록(스텝 1203)을 실행한다. 마지막으로, 무정지 모듈 관리 테이블 및 하드웨어 구성 데이타가 저장되어 있는 재기동 로더(113)의 데이타 영역에 할당되어 있는 페이지를 기록 금지로 설정한다(스텝 1204).

재초기화 루틴은 로드 모듈 관리 테이블로의 인터페이스 루틴의 등록 처리를 한다. 이것은 다른 무정지 모듈의 경우와 마찬가지이다. 하드웨어 구성 데이타 복사 루틴은 재기동 로더(113)의 데이타 영역에 저장되어 있는 하드웨어 구성 데

이타를 커넬(111)의 데이타 영역에 복사한다. 무정지 모듈 등록 루틴은 무정지 모듈에 관한 데이타를 무정지 모듈 관리 테이블(500)에 등록한다. 통상 재기동 로더(113)의 데이타 영역은 가상 어드레스 기구에 의해 기록 금지로 설정되어 있으므로, 등록 루틴은 기록 금지를 일단 해제하여 무정지 모듈 데이타를 테이블(500)에 기록하고 다시 데이타 영역을 기록 금지로 한다.

이 실시 형태에 따르면, 오퍼레이팅 시스템 커넬(111)이 소프트웨어 장애에 의해 정지하여 재기동 처리를 실시해도, 무정지 모듈이 이용한다고 해서 등록한 외부 디바이스의 인터럽트의 처리를 중단하는 일 없이 실시할 수 있다. 또한, 프로세서 리셋트 시에 실행되는 하드웨어 구성 인식 처리를 회피할 수 있으며 재기동까지의 시간을 단축할 수 있다.

다음에 클럭 인터럽트를 계기로 외부 디바이스의 제어를 실시하는 모듈을 무정지 모듈로 하는 실시 형태에 대해 진술한다. 클럭 인터럽트는 통상 커넬(111) 본체가 관리하는 인터럽트이다. 시간 제약이 엄격한 처리를 실시하는 경우에, 클럭 인터럽트를 커넬(111)로 전달되기 전에 끼워 넣어 시간 제약 조건부의 처리를 우선하여 실시하는 모듈을 이용하는 방식이 있다. 인터럽트를 가로채는 인터럽트 핸들러의 설정을 변경하게 되는 것을 말한다. 이러한 시간 제약이 엄한 제어에서는 시간 제약이 엄하면서 동시에 클럭 인터럽트에 대한 처리의 중단이 허용되지 않는 경우가 있다.

본 발명에 따르면, 클럭 인터럽트를 끼워 넣는 모듈을 무정지 모듈로 함으로써, 커넬(111)이 소프트웨어 장애에 의해 정지하여 재기동하게 되어도 클럭 인터럽트에 대한 처리를 중단하는 일 없이 계속할 수 있다.

클럭 인터럽트 처리 모듈을 무정지 모듈로서 등록할 때에 클럭 인터럽트의 인터럽트 번호 및 클럭 인터럽트의 제어 어드레스를 모듈의 점유 자원으로 디바이스로서 무정지 모듈 관리 테이블(500)에 등록한다. 이에 따라, 커넬(111)의 재기동 전후로 클럭 인터럽트의 처리 환경은 계속된다. 또한, 커넬(111)의 재기동 시에 커넬(111)이 자신의 관리 디바이스인 클럭 인터럽트가 다른 모듈에 점유되어 있는 것을 인식하고, 클럭 인터럽트에 관한 설정을 변경하지 않고 기동 처리를 진행시킨다.

이에 따라, 오퍼레이팅 시스템 커넬(111)이 소프트웨어 장애에 의해 정지하여 재기동되어도, 클럭 인터럽트 처리는 중단되는 일 없이 높은 신뢰성 및 높은 이용 가능성의 제어 시스템을 실현할 수 있다.

〈제2 실시 형태〉

본 발명의 제2 실시 형태에 대해 설명한다.

도 13은 이 실시 형태에서의 계산기의 기동 순서에 나타낸 플로우차트이다.

통상의 커넬 초기화 처리인 커넬 데이타의 초기화(스텝 1301), 외부 디바이스의 초기화(스텝 1302) 및 커넬 모듈의 로드와 초기화(스텝 1303)를 실시한다. 모듈의 초기화는 제1 실시 형태에서와 같이 모듈이 무정지 모듈이면 무정지 모듈 관리 테이블(500)에 그 이용 어드레스 정보를 등록한다.

다음 스텝 1304에서는 주기억(102) 중 커넬(111) 및 그 외 모듈이 로드되어 있는 영역에서 무정지 모듈의 데이타 영역 이외의 영역을 자기 디스크(105)의 재기동 파일에 기록한다. 또한, 스텝 1305에서는 페이지 테이블 레지스터 및 커넬 스택의 스택 포인터값을 자기 디스크(105)의 재기동 파일에 보존하고, 스텝 1306에서 다음의 스텝 1307의 어드레스를 자기 디스크(105)의 재기동 파일에 보존한다.

여기서 작성된 재기동 파일은 커넬(111)이 소프트웨어 장애를 위한 재기동 시에 커넬(111)의 소프트웨어 구성을 재현하기 위해서 이용된다. 또, 스텝 1307에서는 초기 프로세스를 작성하여 실행한다.

도 14는 본 발명의 제2 실시 형태에서의 오퍼레이팅 시스템 커넬(111)의 재기동 순서를 나타낸 플로우차트이다.

이 재기동 순서는 무정지 모듈로서 등록되어 있는 재기동 로더(113)에 의해 실시된다. 도면에 따라서 재기동 순서를 설명한다. 우선, 페이지 테이블 레지스터(400)가 가리키고 있는 페이지 테이블(410)을 재기동 로더(113)의 데이타 영역에 복사하고(스텝 1401), 페이지 테이블 레지스터(400)가 복사된 쪽의 페이지 테이블을 가리키도록 변경한다(스텝 1402).

다음에, 자기 디스크(105)의 재기동 파일에 기록된 커넬 기동 시의 주기억(102)의 내용을 주기억(102)에 복사한다(스텝 1403). 계속해서, 페이지 테이블 레지스터와 스택 포인터 및 스텝 1307의 어드레스를 자기 디스크(105)의 재기동 파일에서 취득한다(스텝 1404, 1405). 계속되는 스텝 1406에서 취득한 페이지 레지스터값 및 스택 포인터값을 레지스터에 설정하고, 스텝 1307으로 처리를 이동시킨다(스텝 1407).

제2 실시 형태에서는 자기 디스크(105)의 재기동 파일은 무정지 모듈로서 등록되어 있는 모듈의 데이타 영역을 포함하고 있지 않기 때문에, 커넬 재기동의 전후로 무정지 모듈의 데이타 영역은 주기억(102)에 보존되어 있다. 이에 따라, 무정지 모듈의 외부 인터럽트 처리의 연속성을 유지할 수 있다. 본 실시 형태에서는 주기억(102)의 내용을 디스크(105)에 보존했지만, 주기억의 다른 영역에 보존할 수도 있다.

또한, 커넬(111)이 소프트웨어 장애를 검출할 때, 프로세서의 리셋트를 행하지 않도록 하고, 커넬 재기동중에도 특정한 인터럽트에 대한 처리를 할 수 있도록 인터럽트 처리에 이용되는 주기억의 내용을 주기억 중에 보유한채로, 또한 인터럽트 핸들러의 설정을 주기억 중에 보유한 채로, 커넬(111)의 처리를 정지하도록 할 수도 있다.

발명의 효과

오퍼레이팅 시스템의 커넬(111)이 소프트웨어 장애를 검출하여 재기동할 때에 무정지 모듈 관리 테이블(500)을 참조해서 커넬 재기동시의 외부 디바이스의 초기화 순서 및 로드하는 모듈의 구성을 결정하는 순서를 설정함으로써, 특정한 외부 디바이스가 발생하는 인터럽트에 대한 처리를 커넬(111)의 재기동 처리 중에도 중단하는 일 없이 계속 실시할 수 있다.

또한, 프로세서를 리셋트하는 일 없이 오퍼레이팅 시스템을 재기동함으로써 프로세서 리셋트 시에 실행되는 하드웨어 구성 검사 등의 처리를 회피할 수 있으며 오퍼레이팅 시스템이 재기동까지의 시간을 단축할 수 있다.

(57) 청구의 범위

청구항 1.

커넬과 그 외의 특권 모드에서 실행하는 복수의 로드 모듈로 커넬 공간이 구성되는 오퍼레이팅 시스템을 실행하는 계산기의 재기동 방법에 있어서,

상기 오퍼레이팅 시스템의 로드 모듈중 재기동중에도 인터럽트 처리를 계속시키는 모듈을 미리 지정하는 순서가 있고,

상기 순서로 지정된 로드 모듈은 오퍼레이팅 시스템의 재기동중에는 메모리에 보유되고,

상기 로드 모듈이 처리하는 외부 인터럽트를 프로세서가 수취할 수 있도록 프로세서 및 인터럽트 제어 장치를 설정하고,

상기 로드 모듈 이외의 로드 모듈을 메모리에 로드하여 커넬 공간을 재구축하고,

재기동 처리중 상기 외부 인터럽트가 상기 로드 모듈에 의해 처리되는

것을 특징으로 하는 재기동 방법.

청구항 2.

제1항에 있어서, 상기 재기동은 상기 오퍼레이팅 시스템이 소프트웨어 장애를 받을 때 행해지는 것을 특징으로 하는 재기동 방법.

청구항 3.

제1항에 있어서, 상기 오퍼레이팅 시스템의 재기동중에는 상기 로드 모듈이 존재하는 상기 메모리의 영역을 기록 금지 상태로 하는 것을 특징으로 하는 재기동 방법.

청구항 4.

제1 계산기와 제2 계산기가 네트워크를 통해 접속되는 계산기 시스템에 있어서,

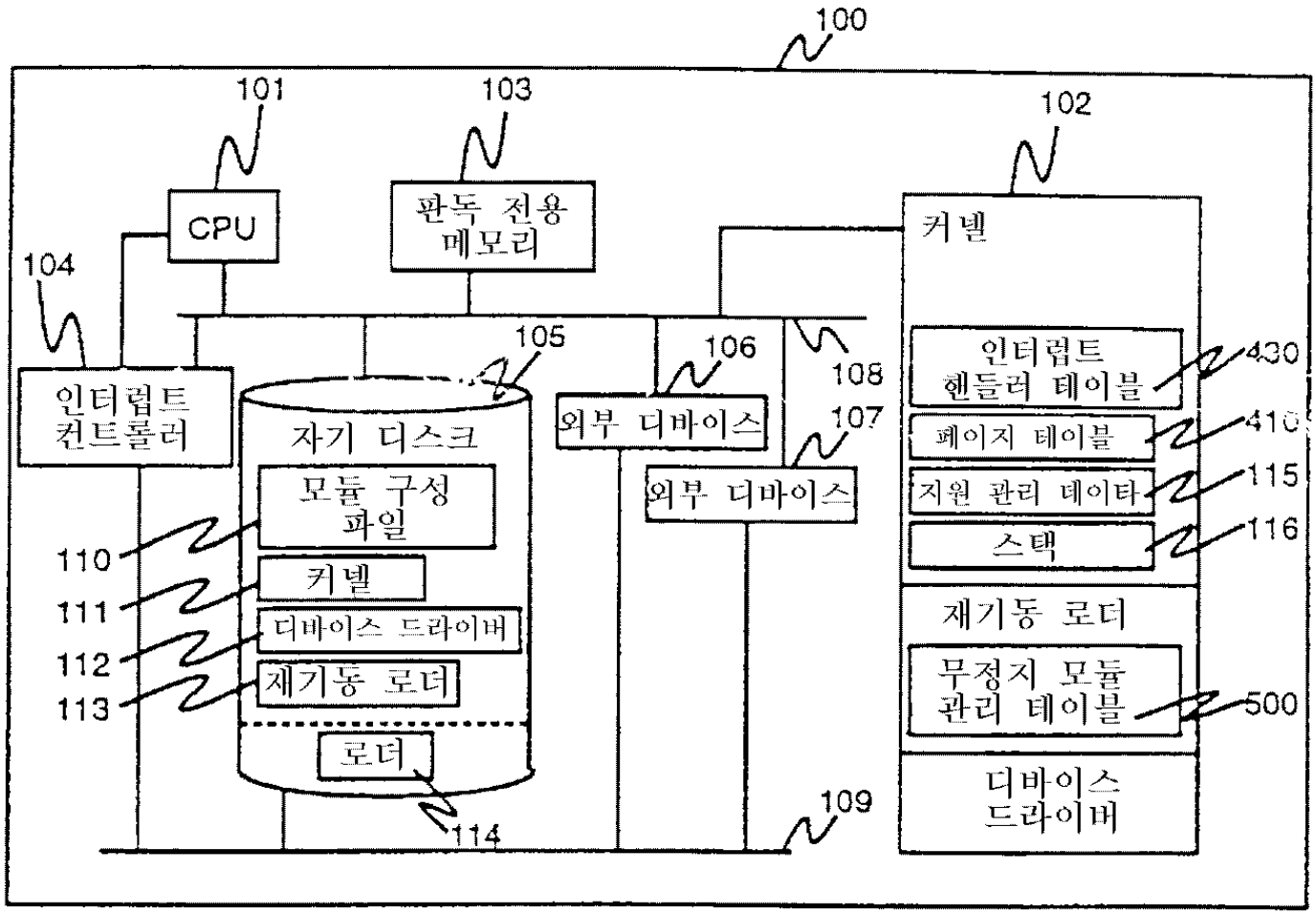
상기 제1 계산기는 상기 제2 계산기에 요청을 보내고,

커넬과 그 외의 특권 모드에서 실행하는 복수의 로드 모듈로 커넬 공간이 구성되는 오퍼레이팅 시스템을 실행하는 상기 제2 계산기는, 상기 오퍼레이팅 시스템의 로드 모듈중 재기동중에도 인터럽트 처리를 계속시키는 모듈을 미리 지정하는 순서가 있고, 상기 순서로 지정된 로드 모듈은 오퍼레이팅 시스템의 재기동중에는 메모리에 보유되고, 상기 로드 모듈이 처리하는 외부 인터럽트를 프로세서가 수취할 수 있도록 프로세서 및 인터럽트 제어 장치를 설정하고, 상기 로드 모듈 이외의 로드 모듈을 메모리에 로드하여 커넬 공간을 재구축하고, 재기동 처리중 상기 외부 인터럽트를 상기 로드 모듈에 의해 처리함으로써 재기동하고,

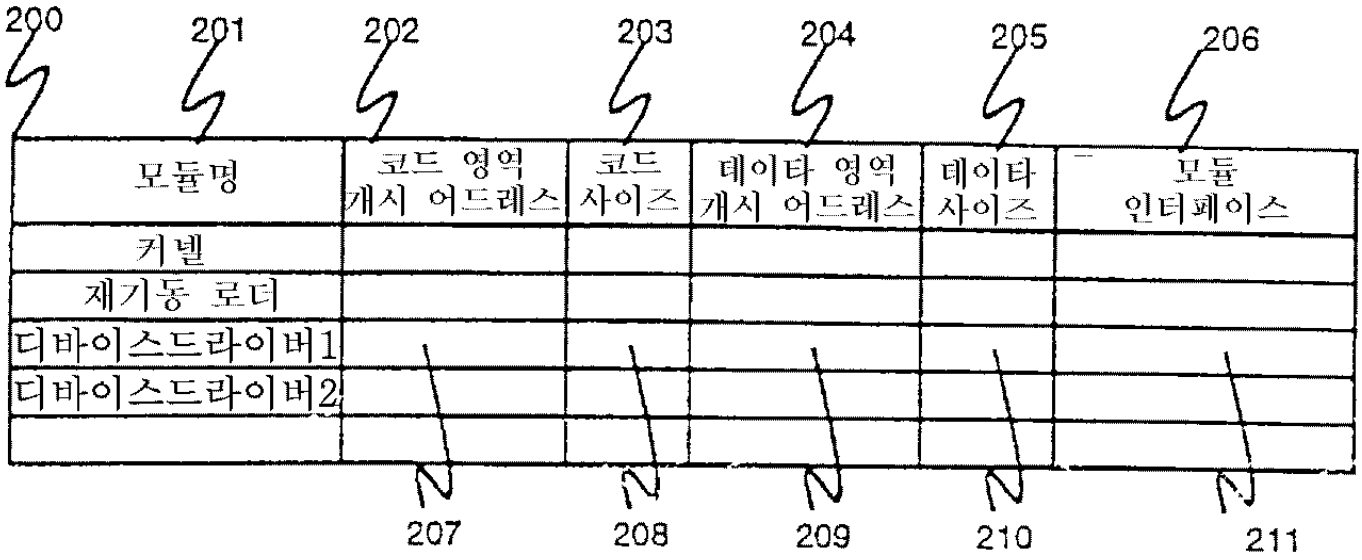
상기 제1 계산기로부터 상기 재기동중에 보내진 상기 요청을 상기 인터럽트에 의해 상기 로드 모듈이 접수하는 것을 특징으로 하는 계산기 시스템.

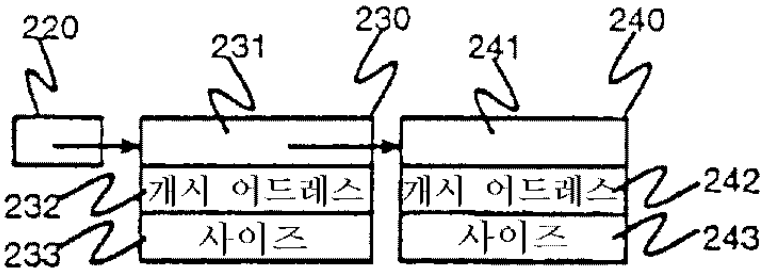
도면

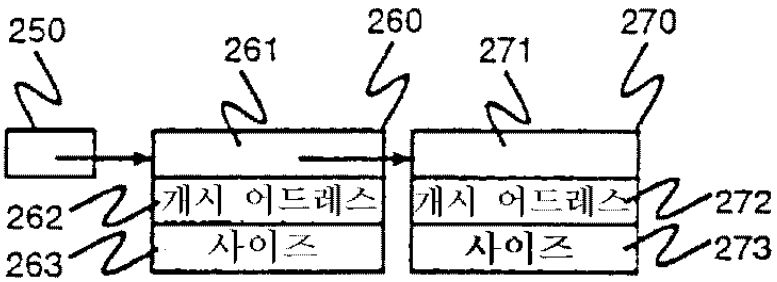
도면 1


100
101
103
102
CPU
판독 전용 메모리
커넬
104
105
106
108
인터럽트 핸들러 테이블
430
인터럽트 컨트롤러
자기 디스크
외부 디바이스
107
페이지 테이블
410
모듈 구성 파일
외부 디바이스
지원 관리 데이타
115
110
커넬
스택
116
111
재기동 로더
디바이스 드라이버
112
무정지 모듈 관리 테이블
500
113
재기동 로더
로더
114
109
디바이스 드라이버

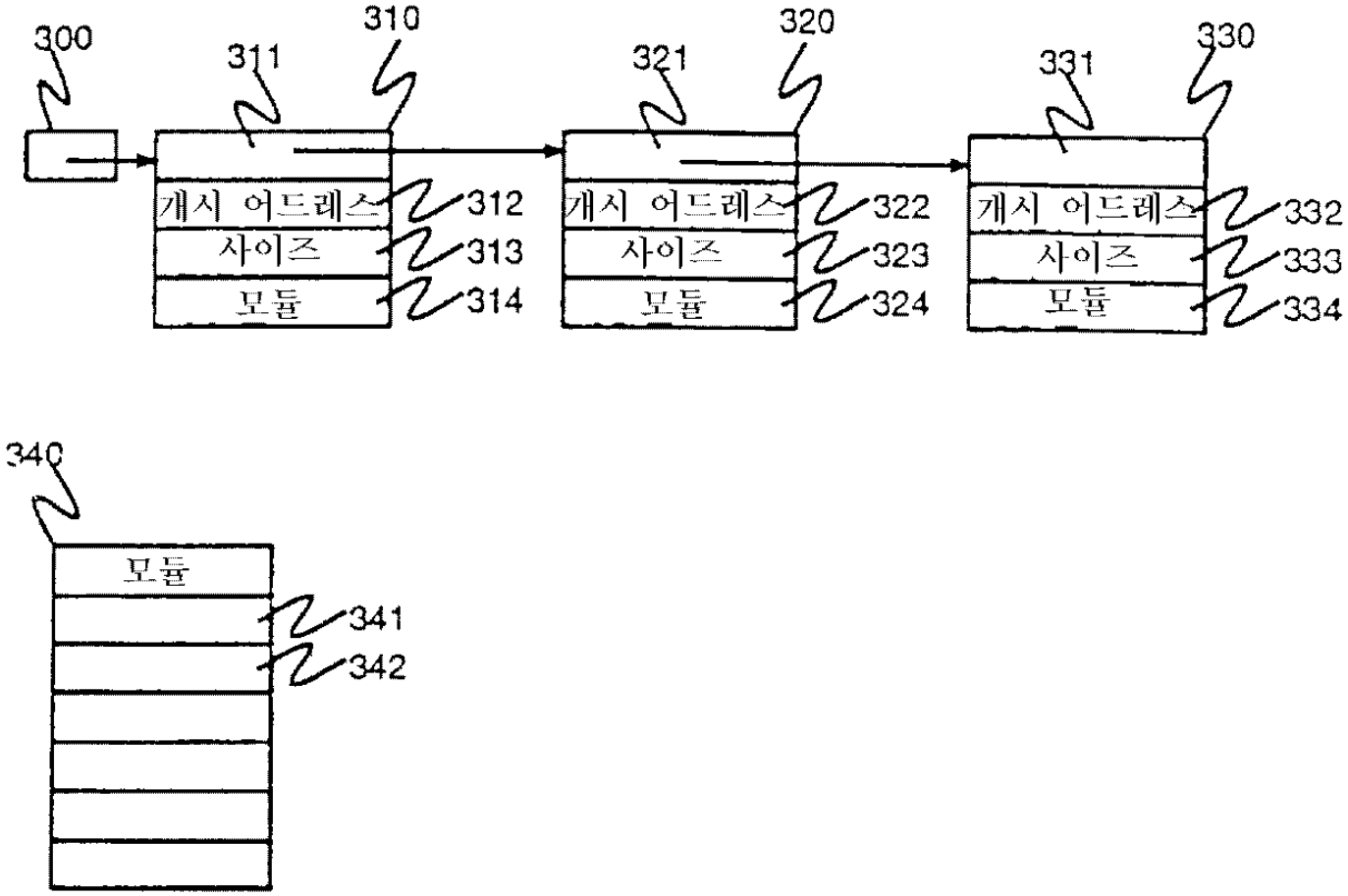
도면 2


200
201
202
203
204
205
206
모듈명
코드 영역 개시 어드레스
코드 사이즈
데이타 영역 개시 어드레스
데이타 사이즈
모듈 인터페이스
커넬
재기동 로더
디바이스드라이버1
디바이스드라이버2
207
208
209
210
211

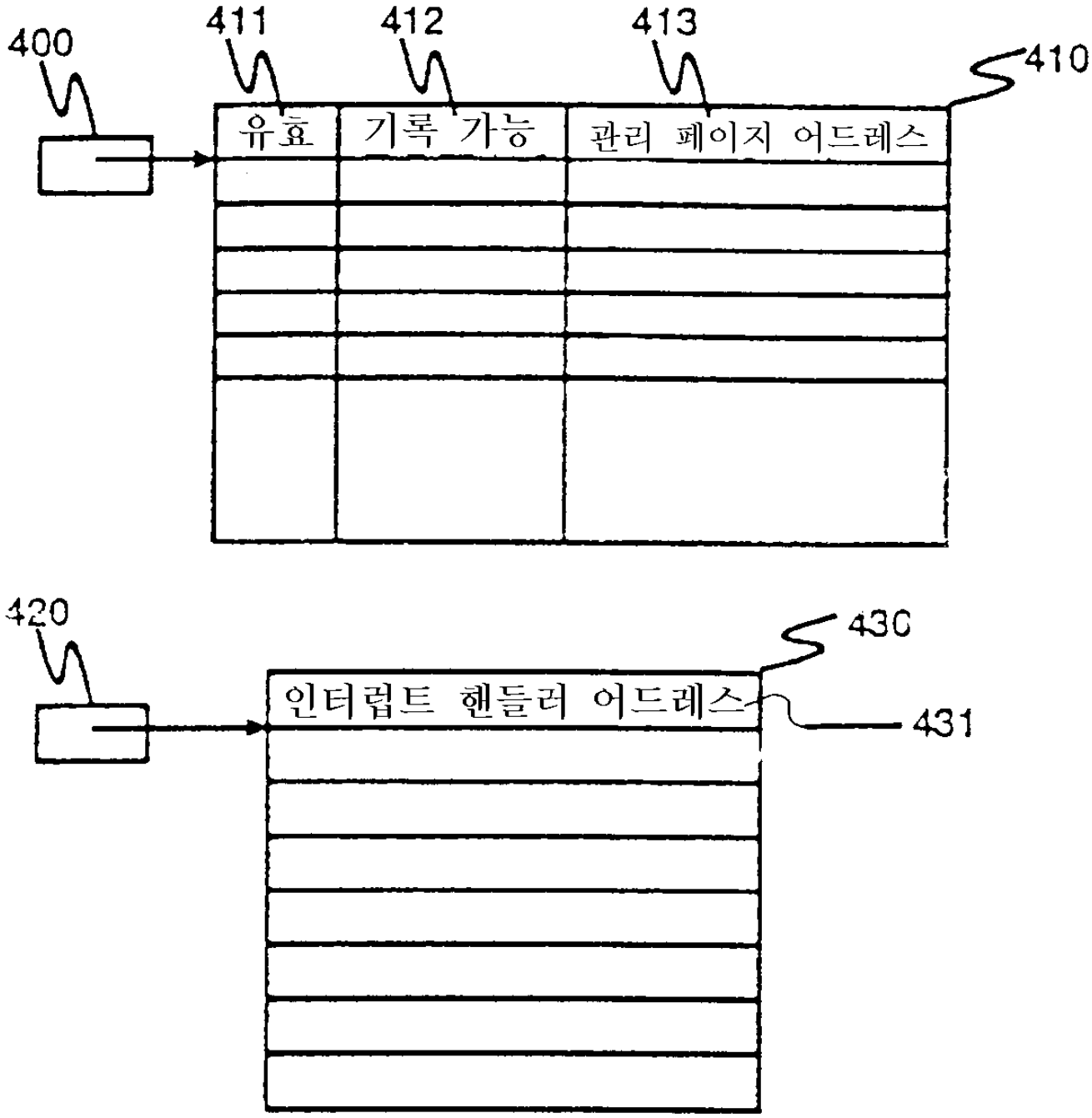

220
231
230
241
240
232
233
게시 어드레스
사이즈
게시 어드레스
사이즈
242
243


250
261
260
271
270
262
263
게시 어드레스
사이즈
게시 어드레스
사이즈
272
273

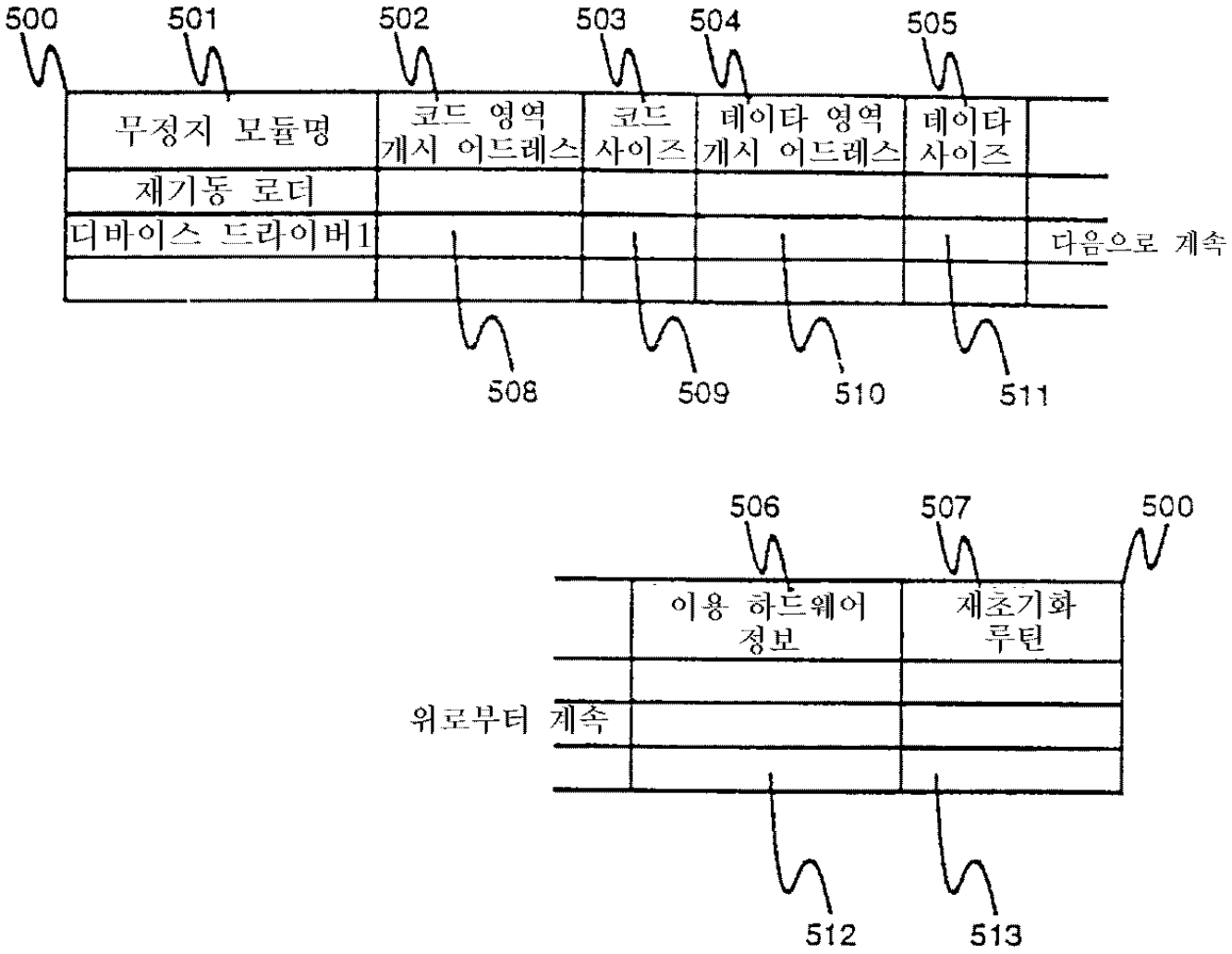
도면 3


300
311
310
321
320
331
330
게시 어드레스
312
사이즈
313
모듈
314
게시 어드레스
322
사이즈
323
모듈
324
게시 어드레스
332
사이즈
333
모듈
334
340
모듈
341
342

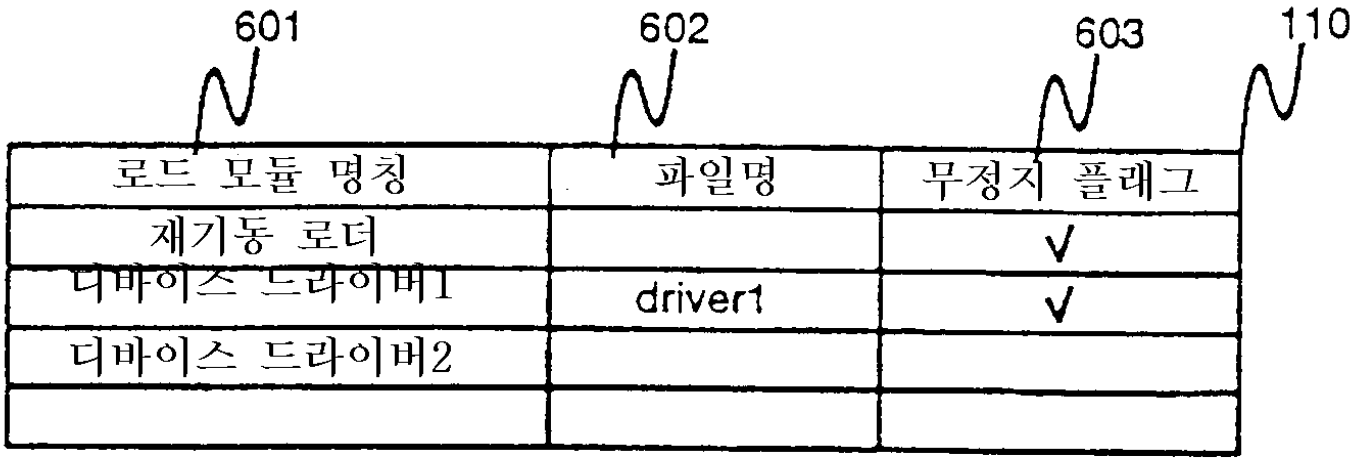
도면 4


400
411
412
413
410
유효
기록 가능
관리 페이지 어드레스
420
430
인터럽트 핸들러 어드레스
431

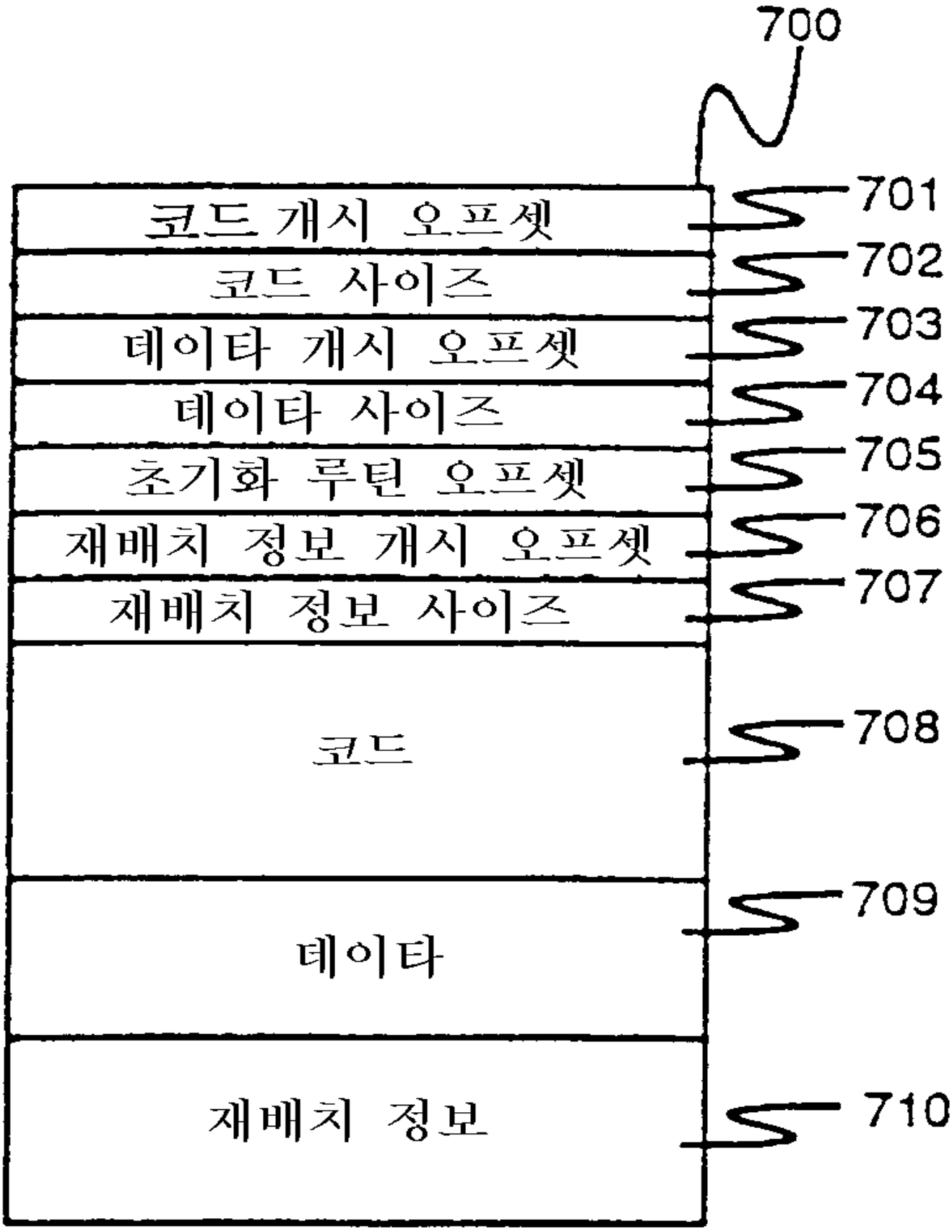
도면 5


500
501
502
503
504
505
무정지 모듈명
코드 영역 게시 어드레스
코드 사이즈
데이타 영역 게시 어드레스
데이타 사이즈
재기동 로더
디바이스 드라이버1
다음으로 계속
508
509
510
511
506
507
500
이용 하드웨어 정보
재초기화 루틴
위로부터 계속
512
513

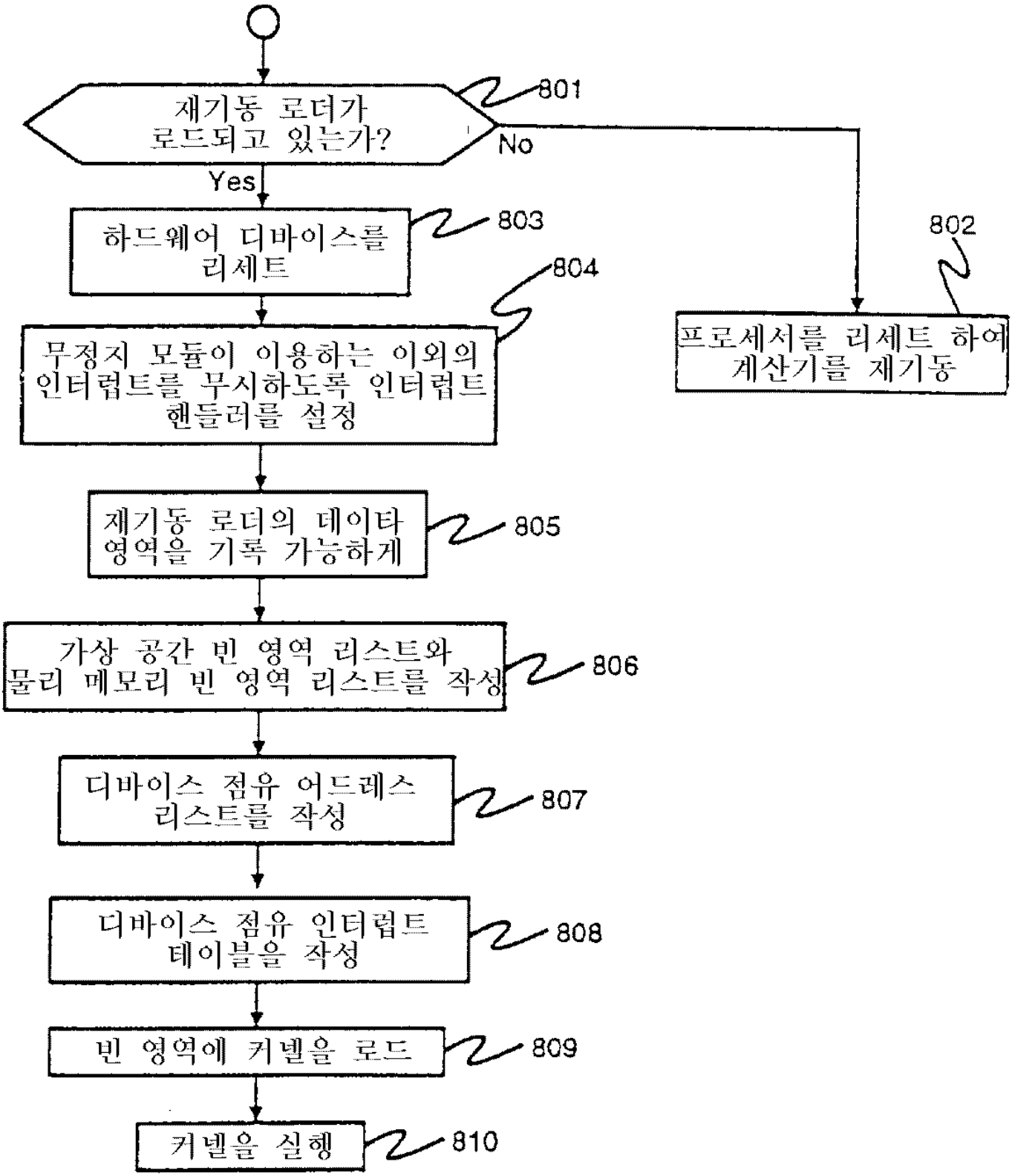
도면 6


601
602
603
110
로드 모듈 명칭
파일명
무정지 플래그
재기동 로더
√
디바이스 드라이버1
driver1
√
디바이스 드라이버2

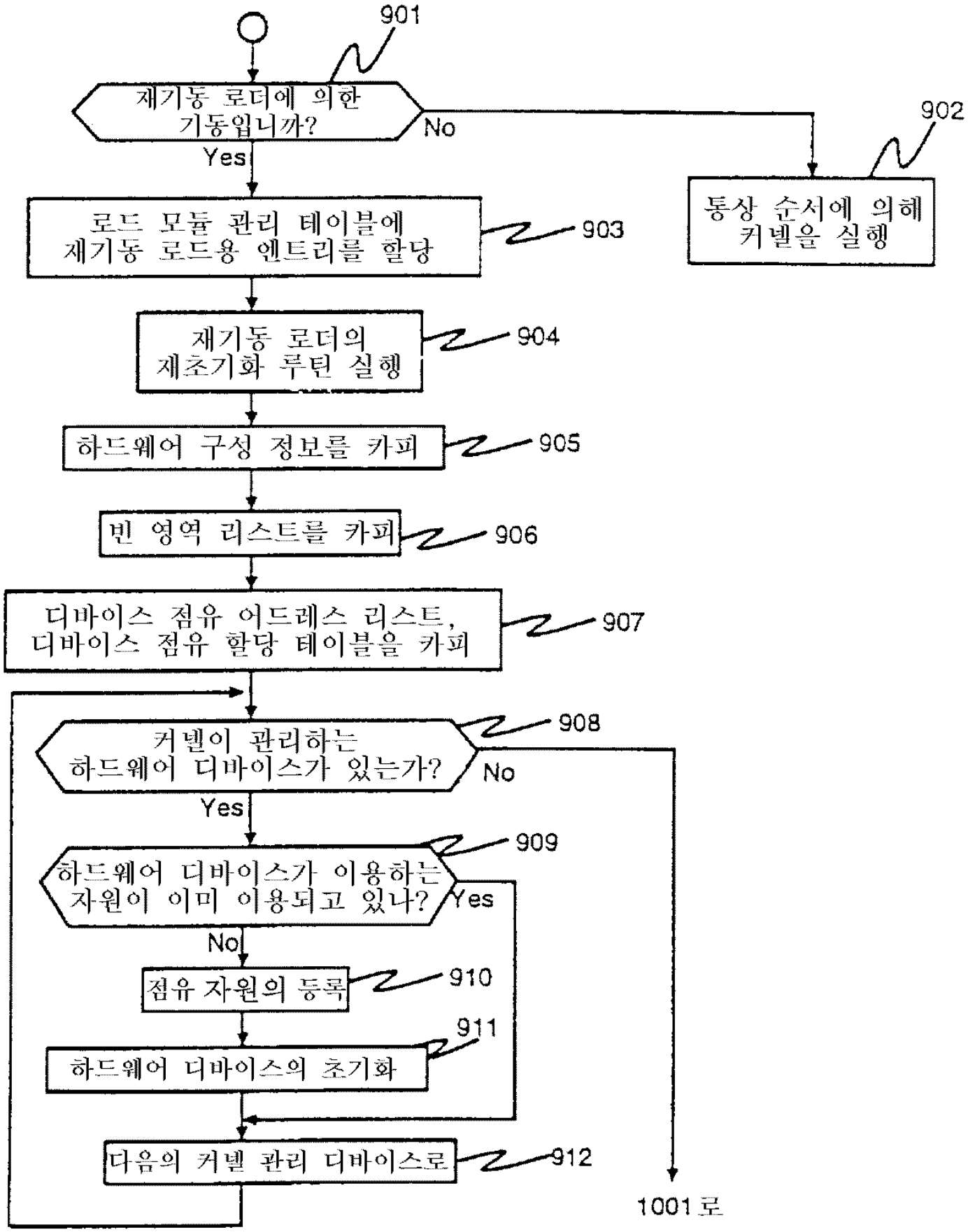
도면 7


700
코드 개시 오프셋
701
코드 사이즈
702
데이타 개시 오프셋
703
데이타 사이즈
704
초기화 루틴 오프셋
705
재배치 정보 개시 오프셋
706
재배치 정보 사이즈
707
코드
708
데이타
709
재배치 정보
710

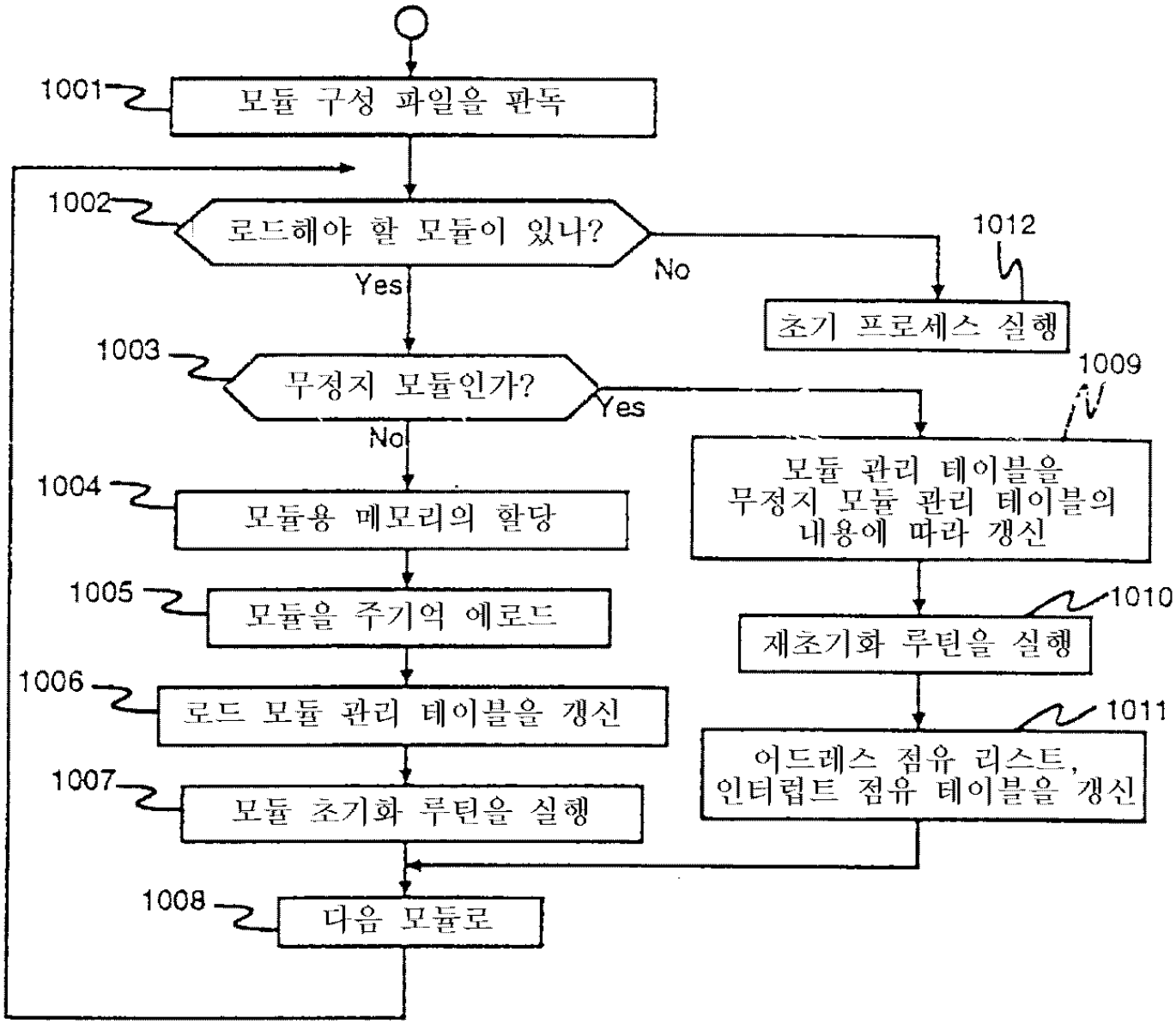
도면 8


재기동 로더가 로드되고 있는가?
801
No
Yes
하드웨어 디바이스를 리세트
803
802
804
프로세서를 리세트 하여 계산기를 재기동
무정지 모듈이 이용하는 이외의 인터럽트를 무시하도록 인터럽트 핸들러를 설정
재기동 로더의 데이타 영역을 기록 가능하게
805
가상 공간 빈 영역 리스트와 물리 메모리 빈 영역 리스트를 작성
806
디바이스 점유 어드레스 리스트를 작성
807
디바이스 점유 인터럽트 테이블을 작성
808
빈 영역에 커넬을 로드
809
커넬을 실행
810

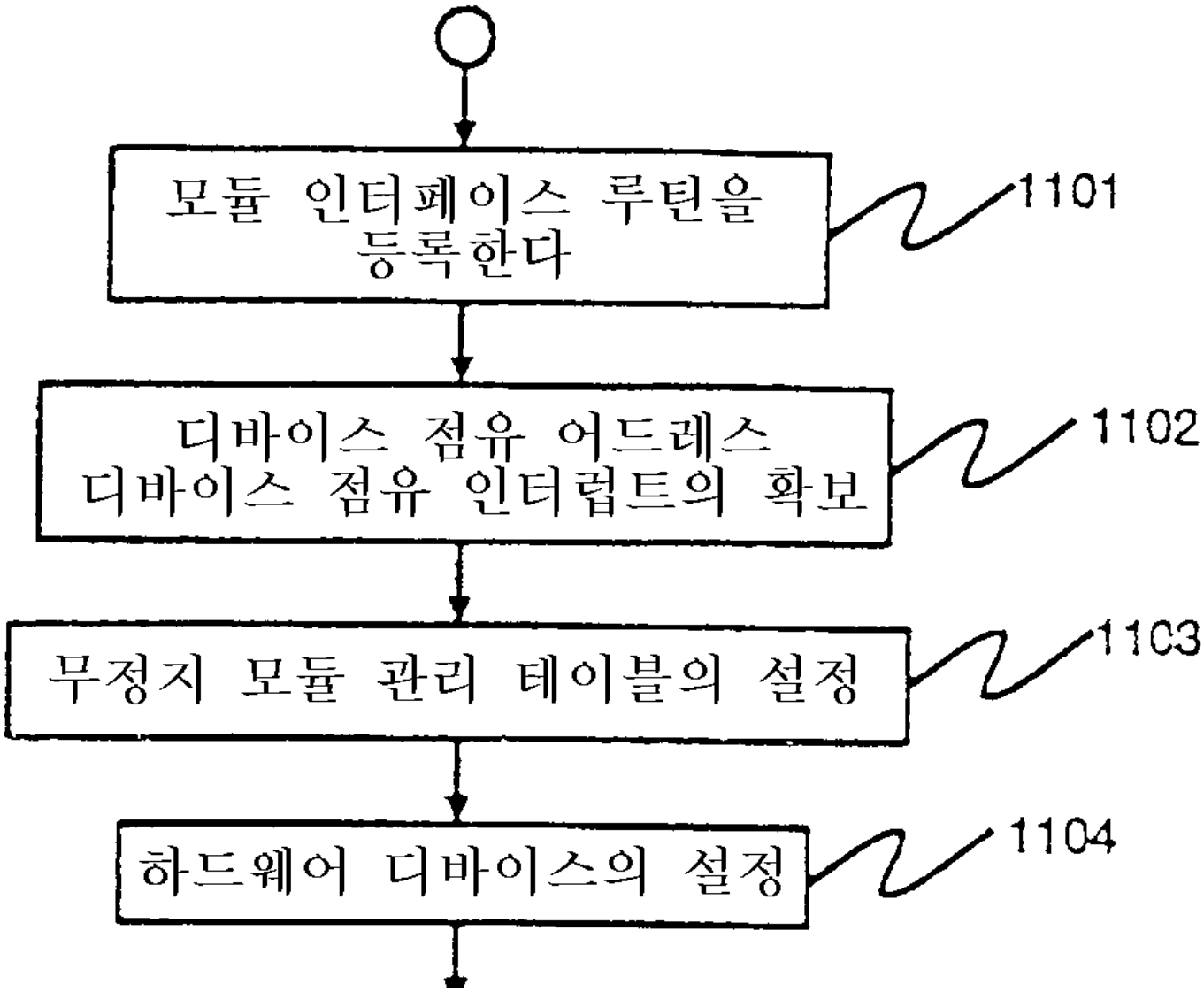
도면 9


901
재기동 로더에 의한 기동입니까?
No
Yes
902
통상 순서에 의해 커넬을 실행
로드 모듈 관리 테이블에 재기동 로드용 엔트리를 할당
903
재기동 로더의 재초기화 루틴 실행
904
하드웨어 구성 정보를 카피
905
빈 영역 리스트를 카피
906
디바이스 점유 어드레스 리스트, 디바이스 점유 할당 테이블을 카피
907
908
커넬이 관리하는 하드웨어 디바이스가 있는가?
No
Yes
909
하드웨어 디바이스가 이용하는 자원이 이미 이용되고 있나?
Yes
No
점유 자원의 등록
910
911
하드웨어 디바이스의 초기화
다음의 커넬 관리 디바이스로
912
1001로

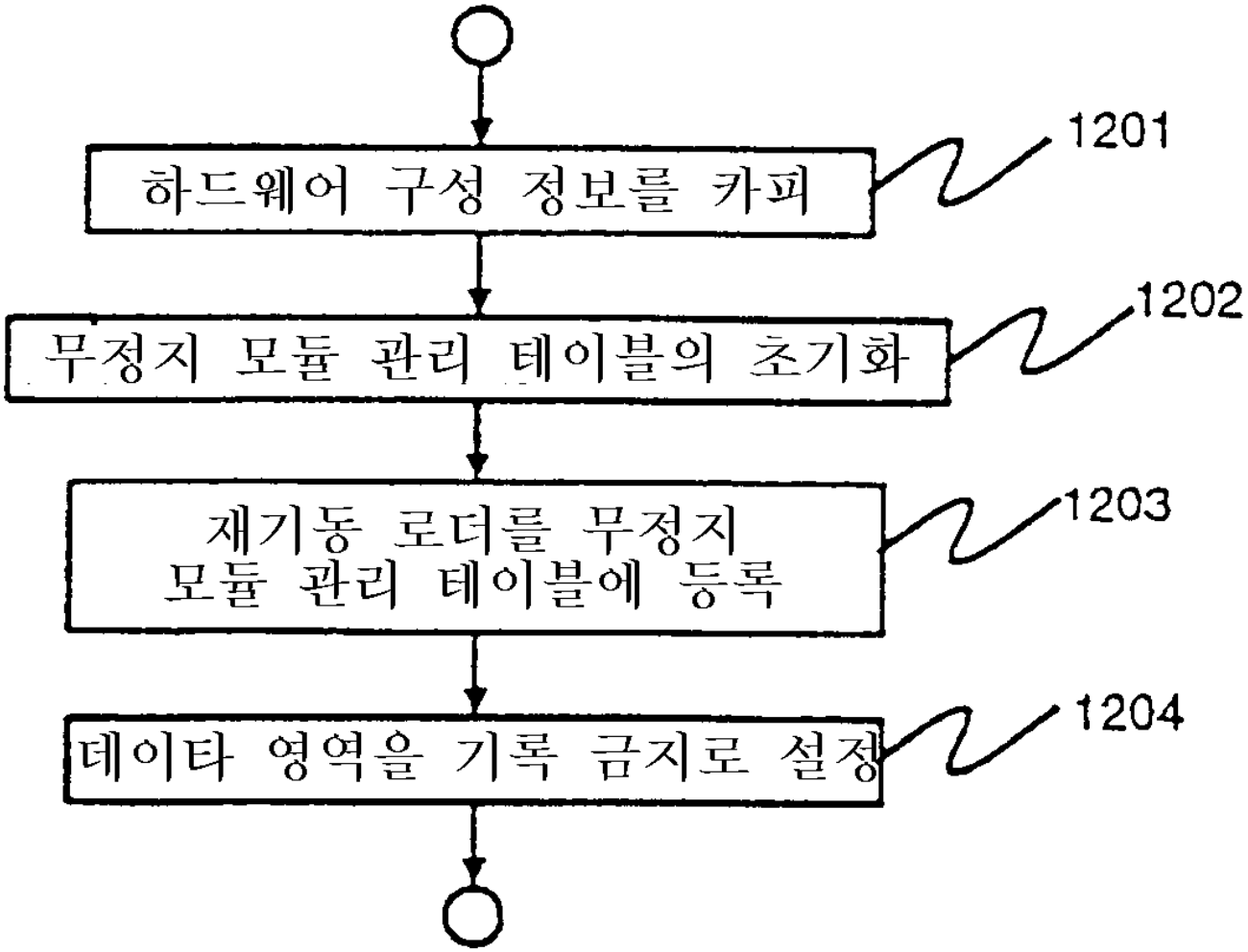
도면 10


1001
모듈 구성 파일을 판독
1002
로드해야 할 모듈이 있나?
No
Yes
1012
초기 프로세스 실행
1003
무정지 모듈인가?
Yes
No
1009
모듈 관리 테이블을
무정지 모듈 관리 테이블의
내용에 따라 갱신
1004
모듈용 메모리의 할당
1005
모듈을 주기억 에로드
1010
재초기화 루틴을 실행
1006
로드 모듈 관리 테이블을 갱신
1011
어드레스 점유 리스트,
인터럽트 점유 테이블을 갱신
1007
모듈 초기화 루틴을 실행
1008
다음 모듈로

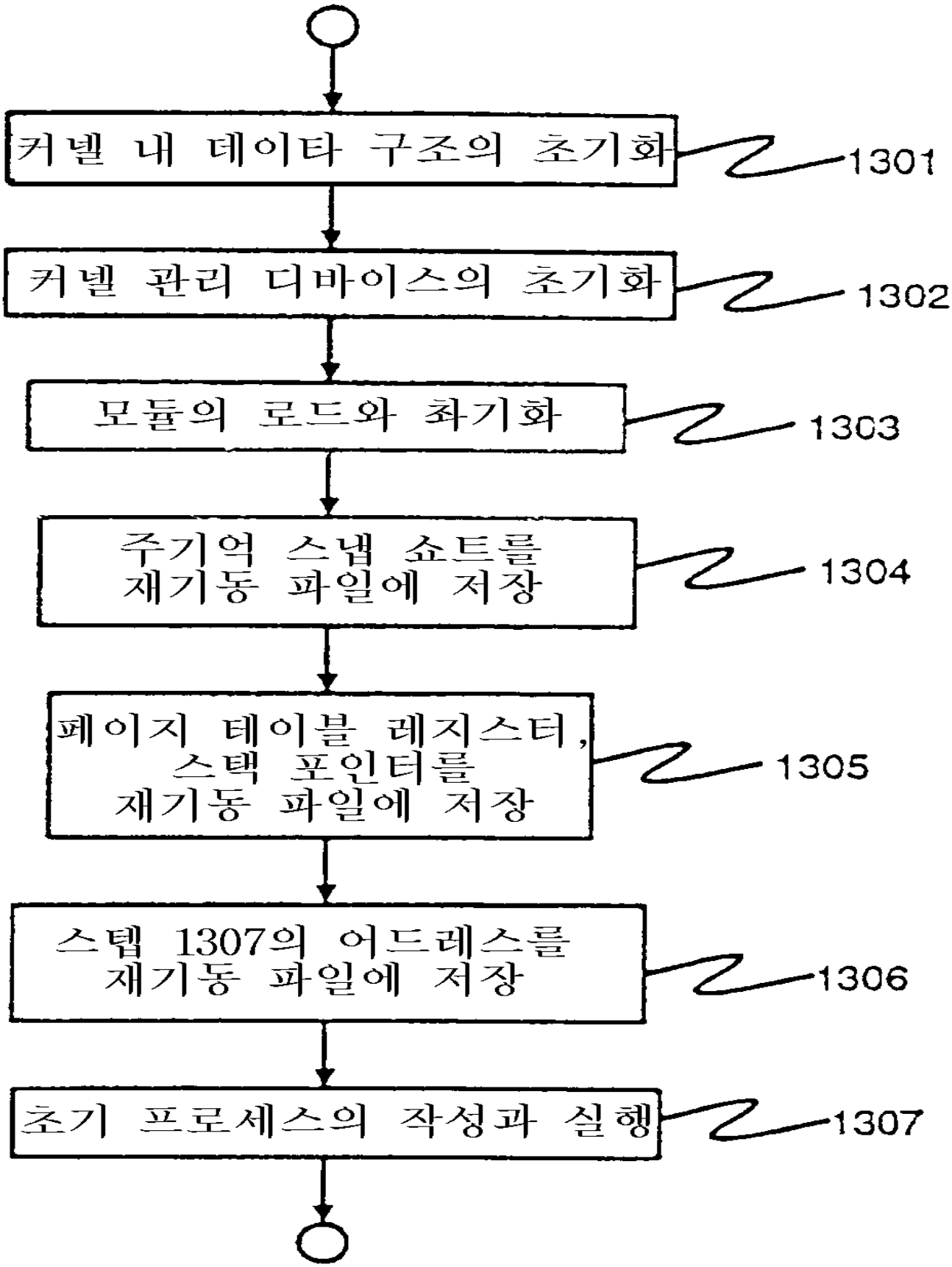
도면 11


모듈 인터페이스 루틴을 등록한다
1101
디바이스 점유 어드레스 디바이스 점유 인터럽트의 확보
1102
무정지 모듈 관리 테이블의 설정
1103
하드웨어 디바이스의 설정
1104

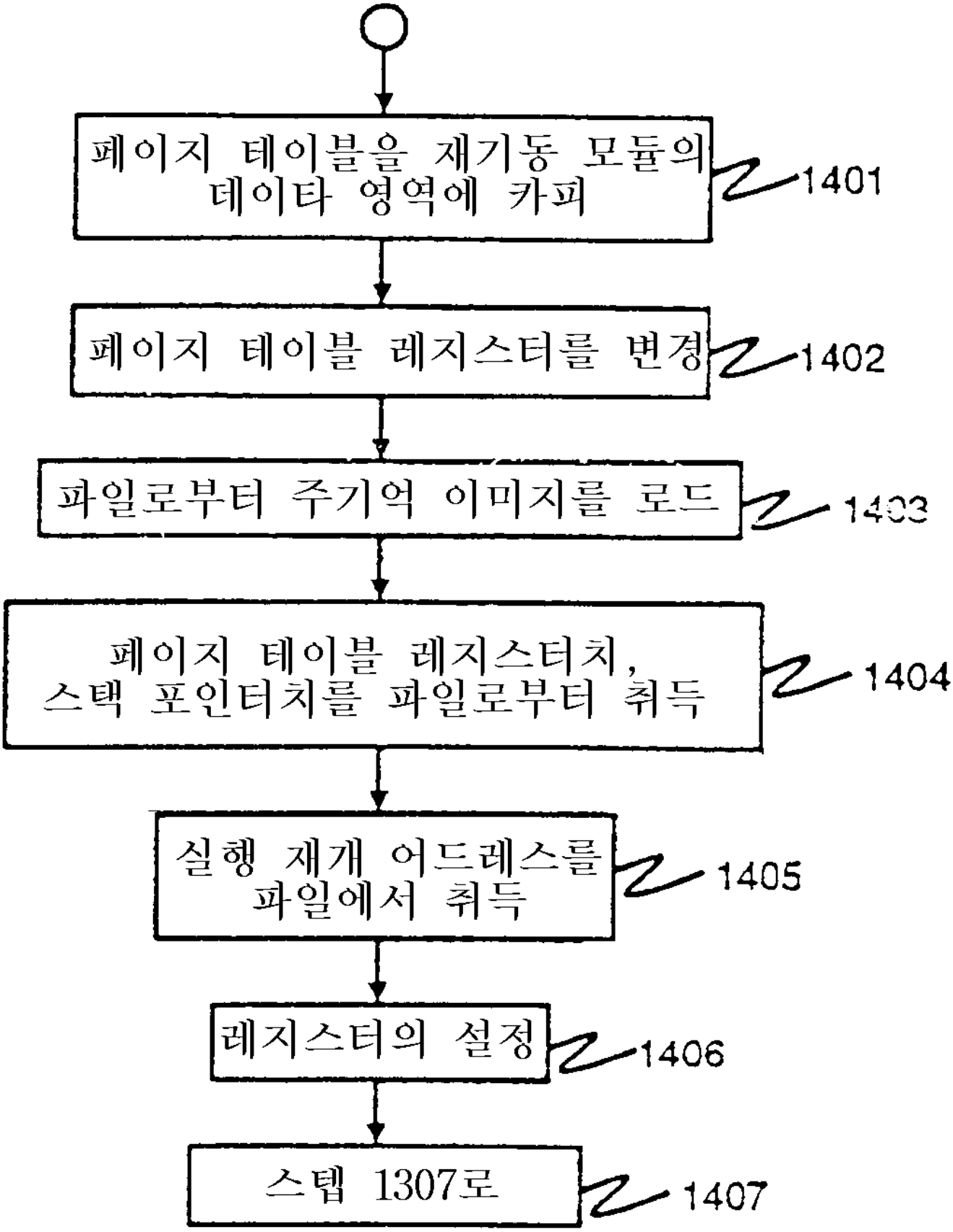
도면 12


하드웨어 구성 정보를 카피
1201
무정지 모듈 관리 테이블의 초기화
1202
재기동 로더를 무정지
모듈 관리 테이블에 등록
1203
데이타 영역을 기록 금지로 설정
1204

도면 13


커넬 내 데이타 구조의 초기화
1301
커넬 관리 디바이스의 초기화
1302
모듈의 로드와 초기화
1303
주기억 스냅 쇼트를
재기동 파일에 저장
1304
페이지 테이블 레지스터,
스택 포인터를
재기동 파일에 저장
1305
스텝 1307의 어드레스를
재기동 파일에 저장
1306
초기 프로세스의 작성과 실행
1307

도면 14


페이지 테이블을 재기동 모듈의 데이타 영역에 카피
1401
페이지 테이블 레지스터를 변경
1402
파일로부터 주기억 이미지를 로드
1403
페이지 테이블 레지스터치, 스택 포인터치를 파일로부터 취득
1404
실행 재개 어드레스를 파일에서 취득
1405
레지스터의 설정
1406
스텝 1307로
1407