(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/46 (2006.01)

(45) 공고일자 2007년02월12일
(11) 등록번호 10-0681875
(24) 등록일자 2007년02월06일

(21) 출원번호	10-2000-7007898	(65) 공개번호	10-2001-0034232
(22) 출원일자	2000년07월19일	(43) 공개일자	2001년04월25일
심사청구일자	2004년01월20일		
번역문 제출일자	2000년07월19일		
(86) 국제출원번호	PCT/FI1999/000039	(87) 국제공개번호	WO 1999/38073
국제출원일자	1999년01월21일	국제공개일자	1999년07월29일

(81) 지정국 국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바베이도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 리히텐슈타인, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬랜드, 일본, 케냐, 키르키즈스탄, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 슬로베니아, 슬로바키아, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 우크라이나, 우간다, 미국, 우즈베키스탄, 베트남, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 가나, 감비아, 크로아티아, 인도네시아, 시에라리온, 짐바브웨, 그라나다, 세르비아 앤 몬테네그로, 인도,

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 가나, 감비아, 짐바브웨,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탄, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 리히텐슈타인, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드, 사이프러스,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고, 기니 비사우,

(30) 우선권주장 980135 1998년01월21일 핀란드(FI)

(73) 특허권자
노키아 모빌 폰즈 리미티드
핀랜드 핀-02150 에스푸 카일알라덴티에 4

(72) 발명자
뢴캐리스토
핀랜드핀-33530탐페레아아릭칼란카투2에이에스.4

사아리넨베사
핀랜드핀-33820탐페레페쿠수온카투21비47

칸톨라잔네
핀랜드핀-33720탐페레엘레멘틴폴쿠15에이6

레스켈래지르키
핀랜드핀-90580오울루주올라베내티에1씨6

렘피넨킴
핀랜드핀-90570오울루텔레르본티에5씨17

푸르호넨아누
핀랜드핀-90800오울루무스타헤룩카티에9이

(74) 대리인 리앤목특허법인

심사관 : 권오성

전체 청구항 수 : 총 34 항

(54) 내장형 시스템

(57) 요약

오퍼레이팅 시스템(OS_A, OS_B)을 구동시키기 위한 적어도 하나의 프로세서(2)를 구비하는 내장형 시스템(embedded system, 1)이 개시된다. 내장형 시스템(1)은, 상기 프로세서(2) 내에서 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)을 구동시키기 위한 수단들(17, 401, 412), 제 1 그룹의 스레드들(THA1, THA2, THA_IDLE)을 구비하는 제 1 오퍼레이팅 시스템(OS_A), 제 2 그룹의 스레드들(THB1, THB2, THB_IDLE)을 구비하는 제 2 오퍼레이팅 시스템(OS_B), 상기 프로세서(2)에 인터럽트(FIQ, IRQ, SWI)를 발생시키기 위한 수단들(nFIQ, nIRQ, SWI), 상기 프로세서(2)에 도달한 인터럽트(FIQ, IRQ, SWI)가 스레드들(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE) 중에 어느 스레드의 실행에 영향을 주는지를 조사(examine)하는 수단들(401, 603, 617) 및 상기 프로세서(2)에 의해 수신된 상기 인터럽트(FIQ, IRQ, SWI)에 영향을 주는 스레드(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE)에 관련되는 상기 오퍼레이팅 시스템(OS_A, OS_B)에 인터럽트 데이터를 전송하기 위한 수단들(401, 412, 603, 609, 617)을 더 구비한다.

대표도

도 4a

특허청구의 범위

청구항 1.

오퍼레이팅 시스템(OS_A, OS_B)을 구동시키기 위한 적어도 하나의 프로세서(2)를 구비하는 내장형 시스템(embedded system, 1)에 있어서, 상기 내장형 시스템(1)은,

- 상기 프로세서(2) 내에서 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)을 구동시키기 위한 수단들(17, 401, 412),

- 제 1 그룹의 스레드들(THA1, THA2, THA_IDLE)을 구비하는 제 1 오퍼레이팅 시스템(OS_A)을 저장하는 수단,

- 제 2 그룹의 스레드들(THB1, THB2, THB_IDLE)을 구비하는 제 2 오퍼레이팅 시스템(OS_B)을 저장하는 수단,

- 상기 프로세서(2)에 인터럽트(FIQ, IRQ, SWI)를 발생시키기 위한 수단들(nFIQ, nIRQ, SWI),

- 상기 프로세서(2)에 도달한 인터럽트(FIQ, IRQ, SWI)가 스레드들(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE) 중에 어느 스레드의 실행에 영향을 주는지를 조사(examine)하는 수단들(401, 603, 617) 및

- 상기 프로세서(2)에 의해 수신된 상기 인터럽트(FIQ, IRQ, SWI)에 영향을 주는 스레드(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE)에 관련되는 상기 오퍼레이팅 시스템(OS_A, OS_B)에 인터럽트 데이터를 전송하기 위한 수단들(401, 412, 603, 609, 617)을 더 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 2.

제 1항에 있어서,

상기 조사 수단들(401, 603, 617)은 상기 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)을 위한 적어도 하나의, 적어도 부분적으로 공통(partially common)인 인터럽트 핸들러(interrupt handler, 603, 617)를 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 3.

제 1항 또는 제 2항에 있어서,

상기 내장형 시스템(1)은 이동국 평션들 및 데이터 프로세싱 평션들을 구비하며,

상기 적어도 두 개의 오퍼레이팅 시스템들(OS_A, OS_B) 중에 제 1 오퍼레이팅 시스템(OS_A)은 이동국 평션들을 실행하는데, 그리고 제 2 오퍼레이팅 시스템(OS_B)은 데이터 프로세싱 평션들을 실행하는데 관련이 있는 것을 특징으로 하는 내장형 시스템.

청구항 4.

제 3항에 있어서,

상기 내장형 시스템(1)은 적어도 부분적으로 이동국 평션들에 관련이 있는 적어도 하나의 사용자 인터페이스(UI1) 및 적어도 부분적으로 데이터 프로세싱 평션들에 관련이 있는 적어도 하나의 사용자 인터페이스(UI2)를 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 5.

제 4항에 있어서,

상기 이동국 평션들에 관련이 있는 상기 사용자 인터페이스(UI1) 및 상기 데이터 프로세싱 평션들에 관련이 있는 상기 사용자 인터페이스(UI2)는 적어도 부분적으로 공통 디스플레이 장치(10, 15)를 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 6.

제 4항 또는 제 5항에 있어서,

상기 이동국 평션들에 관련이 있는 상기 사용자 인터페이스(UI1) 및 상기 데이터 프로세싱 평션들에 관련이 있는 상기 사용자 인터페이스(UI2)는 적어도 부분적으로 공통 데이터 공급 수단(9, 14)을 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 7.

제 1항 또는 제 2항에 있어서,

상기 내장형 시스템(1)은, 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA1, THA2, THA_IDLE) 중 어느 스레드도 실행중이 아닐 경우, 제 1 오퍼레이팅 시스템(OS_A)의 실행으로부터 제 2 오퍼레이팅 시스템(OS_B)의 실행으로 이동하기 위한 수단(2, THA_IDLE)을 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 8.

제 1항 또는 제 2항에 있어서,

상기 내장형 시스템(1)은, 상기 프로세서(2)에 도달한 인터럽트(FIQ, IRQ, SWI)가 제 1 오퍼레이팅 시스템(OS_A)하에서 적어도 하나의 스레드(THA1, THA2, THA_IDLE)를 실행하는데 영향을 주는 경우, 제 2 오퍼레이팅 시스템(OS_B)로부터 이동하여 제 1 오퍼레이팅 시스템(OS_A)을 실행하기 위한 수단(2, SCH_A)을 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 9.

제 1항 또는 제 2항에 있어서,

적어도 상기 제 1 오퍼레이팅 시스템(OS_A)은 실시간 오퍼레이팅 시스템인 것을 특징으로 하는 내장형 시스템.

청구항 10.

제 1항 또는 제 2항에 있어서,

상기 프로세서(2)는 적어도 다음의 모드들:

- 사용자 모드(USER 모드),

- 특권 모드(SVC 모드),

- 미결정 모드(UND 모드), 및

- 하나 또는 다수의 인터럽트 모드들(FIQ 모드, IRQ 모드, SWI 모드)을 구비하며,

상기 제 1 오퍼레이팅 시스템(OS_A)은 적어도 부분적으로 상기 미결정 모드(UND 모드) 내에서 동작하도록 구현되고,

상기 제 2 오퍼레이팅 시스템(OS_B)은 적어도 부분적으로 상기 사용자 모드(USER 모드) 내에서 동작하도록 구현되며, 상기 인터럽트 핸들러(603, 617)는 몇 가지 인터럽트 모드(FIQ 모드, IRQ 모드, SWI 모드) 내에서 동작하도록 구현되는 것을 특징으로 하는 내장형 시스템.

청구항 11.

제 1항 또는 제 2항에 있어서,

상기 제 1 그룹의 스레드들 중에서 어느 하나의 스레드(THA1, THA2, THA_IDLE)가 상기 제 2 오퍼레이팅 시스템(OS_B)을 구비하는 것을 특징으로 하는 내장형 시스템(1).

청구항 12.

제 2항에 있어서,

상기 내장형 시스템(1)은 적어도 하나의 인터럽트(FIQ, IRQ, SWI)를 위한 적어도 두 개의 인터럽트 서비스 루틴 및

상기 적어도 두 개의 인터럽트 서비스 루틴들 중에서 상기 인터럽트(FIQ, IRQ, SWI)와 결합하여 사용될 하나의 인터럽트 서비스 루틴을 선택하는 수단(2)을 구비하는 것을 특징으로 하는 내장형 시스템.

청구항 13.

내장형 시스템(1) 내의 프로세서(2) 내에서 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법에 있어서,

- 상기 프로세서(2) 내에서 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)이 실행되는 단계,

- 상기 제 1 오퍼레이팅 시스템(OS_A)에 관련하여, 제 1 그룹의 스레드들(THA1, THA2, THA_IDLE)이 실행되는 단계,

- 상기 제 2 오퍼레이팅 시스템(OS_B)에 관련하여, 제 2 그룹의 스레드들(THB1, THB2, THB_IDLE)이 실행되는 단계,

- 상기 프로세서(2)에 전달되는 인터럽트(FIQ, IRQ, SWI)가 발생되는 단계,

- 상기 프로세서(2)에 전달된 인터럽트가 상기 스레드(THA1, THA2, THA_IDLE, THB1, THB2, THB_IDLE) 중 어느 스레드의 실행에 영향을 미치는지가 조사되는 단계, 및

- 상기 프로세서(2)에 의해 수신된 인터럽트(FIQ, IRQ, SWI)의 정보가, 상기 프로세서(2)에 의해 수신된 인터럽트(FIQ, IRQ, SWI)에 영향을 미치는 상기 스레드(THA1, THA2, THA_IDLE, THB1, THB2, THB_IDLE)에 관련되는 상기 오퍼레이팅 시스템(OS_A, OS_B)에 전송되는 단계를 구비하는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 14.

제 13항에 있어서,

상기 프로세서(2)에 의하여 수신된 상기 인터럽트(FIQ, IRQ, SWI)를 조사하는 단계에서, 적어도 부분적으로 상기 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)에 공통되는 적어도 하나의 인터럽트 핸들러가 사용되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 15.

제 13항에 있어서,

상기 내장형 시스템(1) 내에서는 이동국 평션들 및 데이터 프로세싱 평션들이 실행되며,

상기 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B) 중에서, 제 1 오퍼레이팅 시스템(OS_A)은 상기 이동국 평션에 관련되며, 상기 제 2 오퍼레이팅 시스템(OS_B)은 상기 데이터 프로세싱 평션들에 관련되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 16.

제 15항에 있어서,

상기 이동국 펑션들은 상기 제 1 사용자 인터페이스(UI1)에 의해 사용되며, 상기 데이터 프로세싱 펑션들은 상기 제 2 사용자 인터럽트(UI2)에 의해 사용되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 17.

제 16항에 있어서,

상기 이동국 펑션들 및 상기 데이터 프로세싱 펑션들은 적어도 부분적으로 공통인 디스플레이 장치(10, 15) 상에 표시되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 18.

제 16항 또는 제 17항에 있어서,

상기 이동국 펑션들 및 상기 데이터 프로세싱 펑션들을 사용하기 위하여, 데이터는 적어도 부분적으로 공통인 수단(9, 14)에 의하여 제공되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 19.

제 13항 내지 제 17항 중 어느 한 항에 있어서,

상기 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA1, THA2, THA_IDLE) 중 어느 스레드도 실행중이 아닐 경우, 상기 제 1 오퍼레이팅 시스템(OS_A)의 실행으로부터 제 2 오퍼레이팅 시스템(OS_B)을 실행하기 위한 변환(transformation)이 이루어지는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 20.

제 13항 내지 제 17항 중 어느 한 항에 있어서,

상기 프로세서(2)에 의해 수신된 인터럽트가 제 1 오퍼레이팅 시스템(OS_A) 하에서 적어도 하나의 스레드(THA1, THA2, THA_IDLE)를 실행하는데 영향을 주는 경우, 제 2 오퍼레이팅 시스템(OS_B)의 실행으로부터 제 1 오퍼레이팅 시스템(OS_A)을 실행하기 위한 변환이 이루어지는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 21.

제 13항 내지 제 17항 중 어느 한 항에 있어서,

적어도 상기 제 1 오퍼레이팅 시스템(OS_A)은 실시간 오퍼레이팅 시스템인 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 22.

제 13항 내지 제 17항 중 어느 한 항에 있어서,

상기 제 1 그룹의 스레드 중 하나의 스레드(THA1, THA2, THA_IDLE) 내에서 상기 제 2 오퍼레이팅 시스템(OS_B)이 실행되는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 23.

제 22항에 있어서,

상기 제 1 그룹의 스레드들의 스레드(THA1, THA2, THA_IDLE) 각각을 위해 결정된 우선 순위가 있으며,

상기 제 2 오퍼레이팅 시스템(OS_B)이 실행되는 스레드의 우선 순위는 상기 제 2 오퍼레이팅 시스템(OS_B)의 실행 시간을 증가시키기 위하여 상승 될 수 있는 것을 특징으로 하는 오퍼레이팅 시스템(OS_A, OS_B)을 실행하기 위한 방법.

청구항 24.

오퍼레이팅 시스템(OS_A, OS_B)을 실행시키기 위한 적어도 하나의 프로세서(2)를 구비하는 통신 장치(communication device, 1)에 있어서,

상기 통신 장치(1)는 :

- 상기 프로세서(2) 내에서 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)을 구동시키기 위한 수단들(17, 401, 412),

- 제 1 그룹의 스레드들(THA1, THA2, THA_IDLE)을 구비하는 제 1 오퍼레이팅 시스템(OS_A)을 저장하는 수단,

- 제 2 그룹의 스레드들(THB1, THB2, THB_IDLE)을 구비하는 제 2 오퍼레이팅 시스템(OS_B)을 저장하는 수단,

- 상기 프로세서(2)에 인터럽트(FIQ, IRQ, SWI)를 발생시키기 위한 수단들(nFIQ, nIRQ, SWI),

- 상기 프로세서(2)에 도달한 인터럽트(FIQ, IRQ, SWI)가 스레드들(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE) 중에 어느 스레드의 실행에 영향을 주는지를 조사(examine)하는 수단들(401, 603, 617) 및

- 상기 프로세서(2)에 의해 수신된 상기 인터럽트(FIQ, IRQ, SWI)에 영향을 주는 스레드(THA1, THA2, THB1, THB2, THA_IDLE, THB_IDLE)에 관련되는 상기 오퍼레이팅 시스템(OS_A, OS_B)에 인터럽트 데이터를 전송하기 위한 수단들(401, 412, 603, 609, 617)을 더 구비하는 것을 특징으로 하는 통신 장치.

청구항 25.

제 24항에 있어서,

상기 조사 수단들(401, 603, 617)은 상기 적어도 두 개의 오퍼레이팅 시스템(OS_A, OS_B)을 위한 적어도 하나의, 적어도 부분적으로 공통(partially common)인 인터럽트 핸들러(interrupt handler, 603, 617)를 구비하는 것을 특징으로 하는 통신 장치.

청구항 26.

제 24항 또는 제 25항에 있어서,

상기 통신 장치(1)는 이동국 평션들 및 데이터 프로세싱 평션들을 구비하며,

상기 적어도 두 개의 오퍼레이팅 시스템들(OS_A, OS_B) 중에 제 1 오퍼레이팅 시스템(OS_A)은 이동국 평션들을 실행하는데, 그리고 제 2 오퍼레이팅 시스템(OS_B)은 데이터 프로세싱 평션들을 실행하는데 관련이 있는 것을 특징으로 하는 통신 장치.

청구항 27.

제 26항에 있어서,

상기 통신 장치(1)는 적어도 부분적으로 이동국 평션들에 관련이 있는 적어도 하나의 사용자 인터페이스(UI1) 및 적어도 부분적으로 데이터 프로세싱 평션들에 관련이 있는 적어도 하나의 사용자 인터페이스(UI2)를 구비하는 것을 특징으로 하는 통신 장치.

청구항 28.

제 27항에 있어서,

상기 이동국 평션들에 관련이 있는 상기 사용자 인터페이스(UI1) 및 상기 데이터 프로세싱 평션들에 관련이 있는 상기 사용자 인터페이스(UI2)는 적어도 부분적으로 공통 디스플레이 장치(10, 15)를 구비하는 것을 특징으로 하는 통신 장치.

청구항 29.

제 27항 또는 제 28항에 있어서,

상기 이동국 평션들에 관련이 있는 상기 사용자 인터페이스(UI1) 및 상기 데이터 프로세싱 평션들에 관련이 있는 상기 사용자 인터페이스(UI2)는 적어도 부분적으로 공통 데이터 공급 수단(9, 14)을 구비하는 것을 특징으로 하는 통신 장치.

청구항 30.

제 24항 또는 제 25항에 있어서,

상기 통신 장치(1)는, 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA1, THA2, THA_IDLE) 중 어느 스레드도 실행중이 아닐 경우, 제 1 오퍼레이팅 시스템(OS_A)의 실행으로부터 제 2 오퍼레이팅 시스템(OS_B)의 실행으로 이동하기 위한 수단(2, THA_IDLE)을 구비하는 것을 특징으로 하는 통신 장치.

청구항 31.

제 24항 또는 제 25항에 있어서,

상기 통신 장치(1)는, 상기 프로세서(2)에 도달한 인터럽트(FIQ, IRQ, SWI)가 제 1 오퍼레이팅 시스템(OS_A) 하에서 적어도 하나의 스레드(THA1, THA2, THA_IDLE)를 실행하는데 영향을 주는 경우, 제 2 오퍼레이팅 시스템(OS_B)로부터 이동하여 제 1 오퍼레이팅 시스템(OS_A)을 실행하기 위한 수단(2, SCH_A)을 구비하는 것을 특징으로 하는 통신 장치.

청구항 32.

제 24항 또는 제 25항에 있어서,

적어도 상기 제 1 오퍼레이팅 시스템(OS_A)은 실시간 오퍼레이팅 시스템인 것을 특징으로 하는 통신 장치.

청구항 33.

제 24항 또는 제 25항에 있어서,

상기 프로세서(2)는 적어도 다음의 모드들:

- 사용자 모드(USER 모드),

- 특권 모드(SVC 모드),

- 미결정 모드(UND 모드), 및

- 하나 또는 다수의 인터럽트 모드들(FIQ 모드, IRQ 모드, SWI 모드)을 구비하며,

상기 제 1 오퍼레이팅 시스템(OS_A)은 적어도 부분적으로 상기 미결정 모드(UND 모드) 내에서 동작하도록 구현되고,

상기 제 2 오퍼레이팅 시스템(OS_B)은 적어도 부분적으로 상기 사용자 모드(USER 모드) 내에서 동작하도록 구현되며, 상기 인터럽트 핸들러는 몇 가지 인터럽트 모드(FIQ 모드, IRQ 모드, SWI 모드) 내에서 동작하도록 구현되는 것을 특징으로 하는 통신 장치.

청구항 34.

제 24항 또는 제 25항에 있어서,

상기 제 1 그룹의 스레드들 중에서 어느 하나의 스레드(THA1, THA2, THA_IDLE)가 상기 제 2 오퍼레이팅 시스템(OS_B)을 구비하는 것을 특징으로 하는 통신 장치.

명세서

기술분야

본 발명은 오퍼레이팅 시스템(operating system)을 실행하기 위한 적어도 하나의 프로세서를 구비하는 내장형 시스템(embedded system)에 관한 것이다. 특히, 본 발명은 내장형 시스템의 프로세서에서 오퍼레이팅 시스템을 실행하기 위한 방법 및 오퍼레이팅 시스템을 실행하기 위한 적어도 하나의 프로세서를 구비하는 통신 장치에 관한 것이다.

배경기술

전자공학 산업에서, 기본적인 동작 구성요소로써 특정 종류의 마이크로 프로세서 또는 이에 상응하는 프로세서를 사용하는 다양한 장치들이 설계되어 왔다. 이 프로세서는 예를 들면, 마이크로 콘트롤러와 결합되어 실장될 수 있는데, 마이크로 프로세서는 동일한 마이크로 칩 내에 배열된 몇 가지 주변 장치를 구비할 수 있다. 또한, 이러한 장치들은 내장형 시스템(embedded system)이라고 불리는데, 내장형 시스템의 예로서는 셀룰러 네트워크 이동 기지국 CMT(Cellular Mobile Telephone), 개인용 컴퓨터(PC, Personal Computer), 개인용 디지털 단말기(PDA, Personal Digital Assistant) 등이 있다. 이러한 장치의 동작을 콘트롤하기 위해서는, 시스템의 타이밍에 맞추어 동작하고, 자원을 콘트롤하고, 다른 프로그램 블록들 간의 메시지의 처리 및 전송이 가능한 기본 동작을 수행할 수 있는 장치에 특수한(device specific) 오퍼레이팅 시스템이 설계되어 왔다. 전형적으로, 이러한 종류의 오퍼레이팅 시스템들은 소위 실시간 오퍼레이팅 시스템(RTOS, real time operating systems)으로 불린다. 이러한 실시간 오퍼레이팅 시스템의 전형적인 특징들을 예를 들면, 키 입력 또는

타이밍과 같은 외부 인터럽트(external interrupt)에 응답하는 응답 시간을 예측하는데 사용될 수 있다는 점, 최소한의 메모리 자원만을 사용한다는 점, 및 이에 덧붙여서 실시간 오퍼레이팅 시스템들이 작업을 콘트롤하는데 있어서 매우 효과적이라는 점 등이 있다. 더 나아가, 실시간 오퍼레이팅 시스템들은 용량(capacity)을 콘트롤할 수 있는 특징(properties)을 갖도록 자주 설계되어 왔다. 용량 콘트롤의 예로는 베터리의 동작 시간을 연장할 수 있는 휴대용 장치가 있다. 이러한 실시간 오퍼레이팅 시스템의 크기는 보통 10kB 및 100kB 사이의 프로그램 코드 및 정보 메모리(RAM, Random Access Memory)가 된다.

비 실시간 오퍼레이팅 시스템(non-real-time operating system)은 예를 들어 상이한 동작을 실행하는데 필요한 응답 시간(response time)이 실시간 오퍼레이팅 시스템의 경우에 비하여 더 길다는 사실에서 실시간 오퍼레이팅 시스템과는 다르다. 또한, 비 실시간 오퍼레이팅 시스템의 경우에는, 응답 시간을 예측할 수도 없다. 비 실시간 오퍼레이팅 시스템에서 멀티태스킹(multitasking)을 수행하면, 각 실행중인 프로세스는 동일한 또는 후 순위의 우선 순위(priority)를 갖는 다른 프로세스의 실행을 중지시키며, 따라서 응답 시간이 길어질 수 있다. 뿐만 아니라, 비 실시간 오퍼레이팅 시스템의 응답 시간은 오퍼레이팅 시스템의 커널(kernel)에서 결정되는 것이 아니므로, 비 실시간 오퍼레이팅 시스템의 실행 속도를 예측하는 것이 힘들어진다.

이동국(mobile station)들은 메모리나 다른 시스템 자원들의 한정된 저장 용량을 갖는다. 더 나아가 이동국 내에서는, 기입가능한 대량 메모리(하드 디스크) 또는 플래시 메모리와 같은 메모리 카드들과 같은 주변 장치를 위한 시스템 지원(system support)이 존재하지 않는 경우가 자주 발생한다. 이동국에서는, 동적 방법으로(dynamic manner) 프로그램을 시스템에 탑재할 수 없다. 즉, 이동국의 소프트웨어는 이동국으로 공정 단계(manufacturing phase)에서 탑재되거나, 또는 어떠한 경우에는 프로그램은 예를 들어 보수(maintenance)하는 작업과 관련해서만 갱신될 수 있다. 덧붙이면, 이동국용 프로그램은 전형적으로 이동국 생산자들에 의하여 생산되고 테스트되기 때문에 많은 이동국들이 메모리 관리 유닛(MMU, Memory Management Unit)을 구비하지 않고 있다. 이러한 이유에 근거하여, 특정 장치에만 호환되는 실시간 오퍼레이팅 시스템들 역시 전술한 특징들을 지원하지 않는다.
개인용 컴퓨터 및 개인용 디지털 단말기를 생산하는 회사들은 전술한 특성을 지원하는 오퍼레이팅 시스템을 개발하여 왔으나, 이러한 오퍼레이팅 시스템들은 실시간으로 동작하지 않으며, 그러므로 특정한 동작을 실행하는데 걸리는 최대 응답 시간(maximum response times)을 예측하는 데에 사용될 수 없다. 이러한 오퍼레이팅 시스템의 크기는 전형적으로는 200kB 내지 수 MB의 범위이며, 따라서 이러한 오퍼레이팅 시스템을 이동국과 같은 휴대용 기계에 탑재하는 것이 용이한 것만은 아니다.

삭제

실시간 오퍼레이팅 시스템 및 특별히 데이터 처리 장치를 위해 개발된 오퍼레이팅 시스템들은 부분적으로 상충되는 시스템 사양을 가진다. 그러므로, 특정한 오퍼레이팅 시스템을 구현함에 있어서, 그 특정한 오퍼레이팅 시스템에 호환 되는 프로그램용 인터페이스(programming interface)를 가지고, 동시에 작은 메모리 소비 및 실시간 동작 특성 등 모든 필요한 특성들을 만족시키도록 구현하는 것은 어렵다.

최근에, 데이터 프로세서 및 원격통신용 장치(telecommunication device)를 결합된 형태로 모두 구비하는 전자 장치들이 개발되어 왔다. 이러한 장치들은 현 사양(specification)에서는 통신용 장치(communication device)라고 불린다. 통신용 장치의 한 예로서는 노키아 9000 통신기(Nokia 9000 Communicator)를 들 수 있는데, 이 노키아 9000 통신기는 상대적으로 작은 크기를 가지며 데이터 프로세싱 동작 및 이동국 동작을 모두 수행한다. 그러나, 이러한 종류의 결합 장치(combined device)들의 하나의 목적은 가급적 종래에 개발된 상품들을 많이 사용하는 것이었다. 그 점에서, 이러한 상이한 장치들을 위한 소프트웨어를 새로운 결합 장치에 호환되도록 유지하는 것에 목적이 있었다. 이러한 목적은 데이터 프로세싱을 수행하는 부분에는 독자적인 오퍼레이팅 시스템을 가지는 독립된 프로세서를 제공하고, 동일한 방법으로 이동국 실시예들은 독자적인 오퍼레이팅 시스템을 가지는 독립된 프로세서를 가지고 있다. 그러므로, 종래 개발된 상품의 결과를 가능한 한 효과적으로 이용하며 새로운 장치의 발매를 가속화하는 것이 가능했다. 그러나, 분리된 프로세서는 일반적으로 단일 프로세서 솔루션(solution)에서 얻을 수 있는 것보다 많은 용량(capacity)을 차지한다. 이 점에 있어서, 이러한 휴대용 장치들에는 더 효율적인 배터리가 필요한데, 만약 그렇지 않으면 단일 프로세서를 사용하는 장치에서 가능한 동작 시간보다 분리된 프로세서를 사용한 장치의 동작 시간은 짧아진다.

종래 기술 솔루션에서, 단일 프로세서 내에서 두 개의 오퍼레이팅 시스템을 실행하는 일은 두 오퍼레이팅 시스템의 특징을 모두 이용하여 오퍼레이팅 시스템을 구현함으로써 가능했다. 이러한 결합 형식이 가지는 문제점은 예를 들면 다음과 같다. 즉, 오퍼레이팅 시스템이 전혀 다른 타입의 특성(property)을 가지면, 결합될 오퍼레이팅 시스템들의 모든 특성을 수행하는 것이 가능하지 않다는 것이다. 이러한 현상은 예를 들면, 각각의 오퍼레이팅 시스템을 위하여 개발된 모든 응용 프로그램을 실행할 수 없다는 문제점, 또는 응용 프로그램을 모두 사용하려면 응용 프로그램을 바꿔야 한다는 문제점들을 야기한

다. 더 나아가, 새로운 응용 프로그램이 개발되면, 하나의 오퍼레이팅 시스템에 기반하여 개발된 응용 프로그램은 다른 오퍼레이팅 시스템에 맞도록 변화되어야 한다. 그러면 응용 프로그램을 변화시키는 작업 이외에 오류가 발생할 확률이 증가한다.

뿐만 아니라, 미국 특허 제 5,278,973호는 단일 프로세서 내에서 다수 개의 오퍼레이팅 시스템이 사용될 수 있는 데이터 프로세서를 개시한다. 그러나, 다수 개의 오퍼레이팅 시스템 중에서 한번에 오직 하나의 오퍼레이팅 시스템 만이 사용될 수 있다. 실행되는 오퍼레이팅 시스템을 바꾸려면, 현재 사용중인 오퍼레이팅 시스템의 수행을 취소(called off)하고 데이터 프로세서가 재시작(reactivated)되어야만 한다.

발명의 상세한 설명

본 발명의 하나의 목적은 단일 프로세서 또는 이와 같은 프로세서에서 적어도 두 개의 오퍼레이팅 시스템의 동작(operation)들이 실행될 수 있도록 하는 내장형 시스템을 제공하는 것이다. 본 발명은 적어도 하나의, 적어도 부분적으로는 공동인(common) 인터럽트 핸들러(interrupt handler)가 구성되어 어떠한 오퍼레이팅 시스템의 스레드(thread) 상에 각 인터럽트가 영향을 끼치는지 조사하도록 하는 아이디어에 기초한다. 여기서, 인터럽트 데이터는 각 오퍼레이팅 시스템으로 전송된다. 본 발명에 의한 내장형 시스템은 프로세서 내에서 적어도 두 개의 오퍼레이팅 시스템을 구동시키기 위한 수단들, 제 1 그룹의 스레드들을 구비하는 제 1 오퍼레이팅 시스템, 제 2 그룹의 스레드을 구비하는 제 2 오퍼레이팅 시스템, 프로세에 인터럽트를 발생시키기 위한 수단들, 프로세서에 도달한 인터럽트가 스레드들 중에 어느 스레드의 실행에 영향을 주는지를 조사(examine)하는 수단들 및 프로세서에 의해 수신된 상기 인터럽트에 영향을 주는 스레드에 관련되는 상기 오퍼레이팅 시스템에 인터럽트 데이터를 전송하기 위한 수단들을 구비하는 것을 특징으로 한다. 본 발명에 의한 내장형 시스템 내의 프로세서 내에서 오퍼레이팅 시스템을 실행하기 위한 방법은, 프로세서 내에서 적어도 두 개의 오퍼레이팅 시스템이 실행되는 단계, 제 1 오퍼레이팅 시스템에 관련하여, 제 1 그룹의 스레드들이 실행되는 단계, 제 2 오퍼레이팅 시스템에 관련하여, 제 2 그룹의 스레드들이 실행되는 단계, 프로세서에 전달되는 인터럽트가 발생되는 단계, 프로세서에 전달된 인터럽트가 상기 스레드 중 어느 스레드의 실행에 영향을 미치는지가 조사되는 단계, 및 프로세서에 의해 수신된 인터럽트의 정보가, 상기 프로세서에 의해 수신된 인터럽트에 영향을 미치는 상기 스레드에 관련되는 상기 오퍼레이팅 시스템에 전송되는 단계를 구비하는 것을 특징으로 한다. 본 발명에 따른, 오퍼레이팅 시스템을 실행시키기 위한 적어도 하나의 프로세서를 구비하는 통신 장치는, 프로세서 내에서 적어도 두 개의 오퍼레이팅 시스템을 구동시키기 위한 수단들, 제 1 그룹의 스레드들을 구비하는 제 1 오퍼레이팅 시스템, 제 2 그룹의 스레드들을 구비하는 제 2 오퍼레이팅 시스템, 프로세서에 인터럽트를 발생시키기 위한 수단들, 프로세서에 도달한 인터럽트가 스레드들중에 어느 스레드의 실행에 영향을 주는지를 조사(examine)하는 수단들 및 프로세서에 의해 수신된 상기 인터럽트에 영향을 주는 스레드에 관련되는 오퍼레이팅 시스템에 인터럽트 데이터를 전송하기 위한 수단들을 더 구비하는 것을 특징으로 한다.

본 발명은 종래 기술의 솔루션에 비하여 상당한 장점을 제공한다. 하나의 프로세서에서 소비되는 용량(capacity)은 분리된 프로세서들이 각 오퍼레이팅 시스템을 위하여 사용되는 경우에 비하면 전형적으로 작다. 덧붙이면, 메모리와 같은 공통 소자들이 더 많이 함께 사용될 수 있기 때문에 면적 요구량이 줄어든다. 본 발명에 의한 내장형 시스템은 응용 프로그램에 변화를 주지 않아도 개발된 응용 프로그램을 사용하는 것을 가능하게 한다. 또한 한 오퍼레이팅 시스템에 기반하여 개발된 응용 프로그램을 다른 오퍼레이팅 시스템에 맞도록 우선 변화시키지 않고서 새로운 응용 프로그램을 각 오퍼레이팅 시스템에 맞도록 개발할 수 있다. 더 나아가, 본 발명은 두 개의 오퍼레이팅 시스템들이 결합되어 각 오퍼레이팅 시스템의 특징을 사용하는 솔루션의 경우 및 각 오퍼레이팅 시스템이 각자의 독립된 처분에 있는 프로세서를 구비하는 솔루션에 비하여 필요한 노동력 및 발생할 수 있는 오류의 양을 줄일 수 있다. 덧붙이면, 본 발명에 의한 솔루션은, 예를 들어 이동국 동작을 위한 실장과 같은 특정한 응용 프로그램 또는 응용 영역(application area)에서 한층 더 최적화될 수 있는 오퍼레이팅 시스템의 모든 특징을 그대로 유지한다.

더 나아가, 소비되는 전력이 감소하기 때문에, 휴대용 장치에서 단일 배터리로 충전해도 더 긴 동작 시간을 얻을 수 있다.

실시예

본 발명을 더욱 잘 이해하기 위하여, 종래 기술에 의한 프로세서 및 오퍼레이팅 시스템 솔루션이 우선 도 1 내지 도 3을 참조하여 설명된다.

명령 세트(instruction set)에 따라서, 프로세서들은 두 분류로 나뉜다. 그것은 완전 명령 세트 컴퓨터(CISC, Complete Instruction Set Computer) 및 축소 명령 세트 컴퓨터(RISC, Reduced Instruction Set Computer)이다. 축소 명령 세트 컴퓨터에서는, 잘 쓰이지 않는 명령어들의 일부분이 삭제되었다. 예를 들면, 어드레싱 형식(addressing form)의 수가 완전

명령 세트 컴퓨터의 경우에 비하여 작을 수 있다. 그러나, 일반적인 응용 프로그램들은 매우 작은 수의 명령들만 사용하기 때문에, 그러한 이유에 의하여 필요한 응용 프로그램을 축소 명령 세트 컴퓨터를 이용하여 실장하는 것이 가능하다. 완전 명령 세트 컴퓨터에 비교하여 축소 명령 세트 컴퓨터의 장점을 예를 들면, 동작 속도가 증가된다는 사실 및 프로그램 코드에 의해 요구되는 메모리 면적이 완전 명령 조합 컴퓨터가 사용될 경우에 비해 작다는 사실이 있다. 특히, 프로그램 메모리의 양(amount)이 감소하면, 장치의 용량 필요량(need of capacity) 역시 감소하고, 그 결과로 장치의 면적 역시 감소될 수 있다.

실용적인 실시예에서는 자주 마이크로 콘트롤러(microcontroller)가 사용되는데, 마이크로 콘트롤러는 실제 프로세서 및 임의 접근 기억 장치(RAM, Random Access Memory), 판독 전용 메모리(ROM, Read Only Memory), 입력/출력 장치들(I/O), 타이머, 아날로그/디지털 변환기(A/D converters), 및 디지털/아날로그 변환기(D/A converters) 등과 같은 마이크로 콘트롤러에 연결된 보조 부품들(auxiliary components)을 구비한다. 그러므로, 이러한 상이한 유니트 사이의 인터페이스 라인들은 가능한 한 작게 만들어질 수 있고, 이 점에서 장치의 크기는 더 감소하게 되며 결과로 신호의 전송 속도가 이러한 상이한 유니트 간에 더욱 증가될 수 있다. 마이크로 콘트롤러 생산자들은 수 개의 상이한 타입의 마이크로 콘트롤러를 생산하는데, 수 개의 상이한 마이크로 콘트롤러들은 동일한 프로세서를 사용하지만 주변 부품들(peripheral components)의 양 및 타입은 바뀔 수 있다. 그러므로, 각 응용 프로그램은 필요하다면 특정한 마이크로 프로세서에 가장 잘 호환되는 형태로 제공될 수 있다.

본 명세서에서 언급된 바와 같이, 마이크로 프로세서 또는 이에 상응하는 프로세서를 사용하는 장치들을 위하여 오퍼레이팅 시스템들은 소위 기초 소프트웨어(basic software)로서 동작하도록 개발된다. 이러한 오퍼레이팅 시스템은 장치의 상이한 유니트들을 콘트롤하고 데이터를 독출하기 위해 사용된다. 또한, 이러한 오퍼레이팅 시스템은 장치 내에 응용 프로그램을 사용할 수 있도록 한다. 개인용 컴퓨터를 예를 들면, 컴퓨터가 켜지면 프로그램 메모리 내에 오퍼레이팅 시스템이 탑재되고, 탑재된 오퍼레이팅 시스템이 특정한 초기화 기능을 수행한다. 계속적으로, 사용자는 원하는 응용 프로그램을 시작하려 하면, 실행된 오퍼레이팅 시스템이 사용자가 내린 명령을 접수(receive)하고, 원하는 응용 프로그램을 찾고(search), 찾은 응용 프로그램을 프로그램 메모리에 탑재(load)하고, 그 응용 프로그램을 실행한다. 표면적으로 동시에(ostensibly simultaneously) 이러한 응용 프로그램 수 개를 사용하는 것이 가능한데, 이 경우에 응용 프로그램이 이렇게 상이한 응용 프로그램들 간에 실행 시간을 제공하고 스케줄링을 수행한다. 첨가하면, 오퍼레이팅 시스템은 이러한 상이한 응용 프로그램 간에 신호를 전송하고 필요할 경우 응용 프로그램 및 주변 장치 간의 신호를 배열(arrange)한다.
오퍼레이팅 시스템은 소위 스레드(thread) 처럼 프로세스의 실행을 콘트롤할 수 있다. 하나의 스레드는 하나 또는 수 개의 프로세스들의 실행과 관련한 펑션(function)들을 구비할 수 있고, 다시 말하면, 하나의 프로세스는 하나 또는 그 이상의 스레드로 분리될 수 있다. 스레드를 실행하는 것은 오퍼레이팅 시스템의 스케줄러(scheduler)에 의하여 콘트롤된다. 다시 말하면, 스레드는 오퍼레이팅 시스템에 의하여 프로세스에 주어진 일종의 지원 요소(support)이다.

삭제

알려진 오퍼레이팅 시스템들에서는 다음의 특징(property)들이 발견되는데 그 특징들은 코맨드 인터프리터(command interpreter) 및 스케줄러들이다. 코맨드 인터프리터는 예를 들어 코맨드 내에서 응용 프로그램의 명칭을 독출하고 및 사용자에 의하여 제공되고 응용 프로그램에 의하여 전송되어야할 파라미터(parameter)들을 인식하는 등 사용자에 의하여 주어진 코맨드들을 해독한다. 스케줄러는 상이한 프로세스에 자원(resource)들을 배치하는 작업 및 예를 들어 인터럽트에 반응하는 것과 인터럽트 리퀘스트(interrupt request) 신호를 오퍼레이팅 시스템에 전송하는 것과 같이 인터럽트에 의해 요구되는 수단(measure)들의 실행 작업을 수행하는 인터럽트 리퀘스트 서비스(interrupt request service)에 종사한다. 덧붙이면, 오퍼레이팅 시스템은 키패드를 읽거나, 디스플레이(display)에 기입하거나, 가능한 외부 인터페이스 등과 같은 작업을 위한 펑션 블록(functional block)을 구비한다. 자원 중에는 메인 메모리, 주변 장치, 및 프로세서 시간의 사용(use of processor time) 등이 포함된다. 일 예를 들면, 오퍼레이팅 시스템은 실행될 각 프로세스를 위한 메모리 영역을 확보하는데, 실행되는 프로세스는 이 영역에 데이터를 기입하고 이 영역으로부터 데이터를 독출할 수 있다. 어떤 프로세서가 프로세스를 위하여 예약된 메모리 영역 이외의 메모리에 기입하려고 시도하는 경우에, 메모리 콘트롤 유니트(memory control unit)는 이러한 기입이 일어나지 않도록 방지하고 전형적으로는 이러한 프로세스가 수행되는 것을 중지시킨다.

현재 사용되는 많은 프로세서들은 여전히 상이한 모드들을 가진다. 상이한 모드들에는 사용자 모드 및 특권 모드(privileged mode) 또는 관리자 모드(supervisory mode) 또는 커널 모드(kernel mode) 등이 있다. 사용자에 의하여 실행된 응용 프로그램 및 다른 프로세스들은 일반적으로 사용자 모드에 따라 동작하게 된다. 반대로, 오퍼레이팅 시스템이나 오퍼레이팅 시스템에 의해 실행된 프로세스들 중 일부는 전형적으로 특권 모드에서 실행된다. 사용자 모드는 동작 가능성에 있어서 더욱 한정적인데 그 예를 들면, 프로세서의 코맨드들 중 일부는 특권 모드에서만 실행되고 사용자 모드에서는 사용될 수 없다는 점이 있다. 어떠한 오퍼레이팅 시스템에서 특권 동작 모드는 수 개의 레벨을 가지는데, 그 예를 들면 세

개의 단계가 있다. 동작 모드의 커널은 특권 모드 중에서 첫 번째 레벨 내에 실장되고(implemented), 이 커널은 가장 넓은 권리를 갖는다. 두 번째 레벨은 예를 들어 콘트롤 루틴(control routine) 및 이와 같은 것(실행 루틴, Executive) 등을 가진다. 세 번째 레벨은 예를 들어 코맨드 해독기와 같은 것을 포함한다.

프로세서(2)는 일반적으로 인터럽트를 사용할 수 있는 가능성을 가진다. 프로세서(2)의 외부에서 인터럽트 리퀘스트(interrupt request)가 하나 또는 수 개의 인터럽트 서비스 라인(nIRQ, nFIQ)을 통해서 전송될 수 있다. 예를 들어, 로직 모드 1에서 로직 모드 0으로의 변화와 같은 모드 내의 변화가 발생하면, 인터럽트가 프로세서(2)에 전달된다. 프로세서(2)에 주어지는 인터럽트 리퀘스트는 프로세서(2)가 개별적인 인터럽트 서비스 루틴(respective interrupt service routine)을 실행하게 하는데, 개별 인터럽트는 실질적으로 지연이 없이 실행되거나 더 높은 우선 순위를 가지는 인터럽트 리퀘스트가 대기하지 않을 경우에 실행된다. 인터럽트 서비스 루틴 내에서는 적어도 일부의 인터럽트 핸들링 동작이 실행된다. 이러한 핸들링 동작은 예를 들어 개별 응용 프로세서 및 인터럽트의 발생 원인에 의존한다. 인터럽트 서비스 루틴의 동작이 실행 중인 프로세서 및 오퍼레이팅 시스템의 동작 속도를 감속시키기 때문에, 특히 실시간 오퍼레이팅 시스템에서 인터럽트 서비스 루틴을 가능한 한 짧고 빠르게 만들도록 하는 시도가 이루어졌다. 그러므로, 인터럽트 서비스 루틴 내에서는 오퍼레이팅 시스템을 위하여 인터럽트를 알리고 및 인터럽트 핸들링 수단(interrupt handling measures)이 실행을 대기하고 있다는 정보를 알리는 상태 변수(state variable)의 값을 설정하는 것이 가능한데, 그러면 오퍼레이팅 시스템은 이러한 동작이 실행되는 것을 예를 들어 자신의 스케줄링 프로시져(scheduling procedure)에 따라서 콘트롤한다. 인터럽트 서비스 루틴으로 전송하는 작업은 다양한 인터럽트 차단(blocking) 및 마스킹(masking)에 의해 영향받는다. 인터럽트들이 비활성화(disable) 되면 인터럽트 리퀘스트는 일반적으로 인터럽트 차단이 취소될 때까지 대기하는데, 그 이후에 인터럽트 리퀘스트는 그들의 우선 순위에 따라서 우선적으로 제공된다. 인터럽트들은 거의 모든 프로세서(2) 내에서 선택적으로 방지될 수 있는데, 이 점에서 인터럽트 마스크 레지스터(interrupt mask register) 또는 이와 유사한 레지스터 내에서 각 인터럽트들이 활성화 또는 비활성화 상태(state)를 가지고 제공된다.

인터럽트 리퀘스트들이 발생되는 경우의 예를 들면, 외부 장치(external device)에 의하여 주어지는 신호에 근거하여 발생되는데, 외부 장치의 예를 들면 시리얼 버스(serial bus)를 통하여 수신된 정보에 의하여 야기되는 시리얼 버스 인터럽트 리퀘스트, 키패드에서 키를 누름으로써 발생되는 인터럽트 리퀘스트, 타이머에 의하여 발생되는 인터럽트 리퀘스트, 또는 몇 가지 실행 프로세스에서 발생되는 소위 프로그램 인터럽트 리퀘스트가 있다. 인터럽트들에 대하여 우선 순위가 정해질 수 있는데, 낮은 순위 분류(lower priority classification)를 가지는 인터럽트 서비스 루틴의 실행은 높은 우선 순위 분류를 가지는 인터럽트에 의하여 정지될 수 있다. 반대로, 높은 우선 순위 분류를 가지는 인터럽트 서비스 루틴의 실행은 낮은 순위 분류를 가지는 인터럽트에 의하여 일반적으로는 정지되지 않는다. 오퍼레이팅 시스템을 위한 기본 타이밍을 얻기 위하여 일반적으로 타이머가 사용되는데, 이 타이머는 특정 간격으로 인터럽트 리퀘스트를 발생하여 프로세서에 전송하고, 그러면 프로세서가 변화되어 타이머의 인터럽트 서비스 루틴을 수행한다. 결과적으로, 타이머의 인터럽트 서비스에는 높은 우선 순위가 주어질 가능성이 높다. 반면에, 예를 들어 디스플레이를 갱신하는 작업에는 일반적으로 낮은 우선 순위가 부여된다.

미국 특허 번호 제 5,515,538호는 데이터 프로세서의 멀티프로세싱 (multiprocessing) 오퍼레이팅 시스템 내에서 인터럽트를 핸들링하는 하나의 방법을 개시한다. 이 방법에서, 인터럽트 핸들러가 구현되어 독자적인 스레드를 구성하는데, 인터럽트 상황(interruption situation)에서 실행 코맨드가 직접 이 스레드로 전달된다. 그러나, 이 특허에서 소개된 방법은, 이 오퍼레이팅 시스템의 프로세스 들을 실행하기 위한, 동시에 수 개의 프로세서를 포함할 수 있는 단일 오퍼레이팅 시스템의 장치에 대해서만 언급하고 있다.

이하, 단일 멀티프로세싱 오퍼레이팅 시스템의 동작에 대해서 간략히 설명한다. 오퍼레이팅 시스템의 스케줄러는 특정 간격으로 어떠한 응용 프로그램(즉, 프로세스)이 그 순간에 실행되어야 하는지 조사한다. 이 스케줄러는 예를 들면 타이머 인터럽트를 이용하여 기동된다. 각 응용 프로그램은 더 나아가 단일 또는 수 개의 스레드를 구비할 수 있는데, 각 스레드는 상이한 시간에 단일 프로세서 시스템 내에서 수행된다. 동작의 예를 들면, 하나의 스레드는 키패드와 같은 주변 인터페이스 상의 데이터를 판독하고, 두 번째 스레드는 그 데이터를 처리하며, 세 번째 스레드는 예를 들어 디스플레이와 같은 주변 장치 상에 처리된 데이터를 표시할 수 있다. 이 응용 프로그램이 시작되면, 스케줄러는 이 스레드를 실행하기 위한 반복 간격(repetition interval) 뿐만 아니라 한번에 하나의 스레드를 위한 동작 시간을 결정한다. 그러므로, 스케줄러는 그 스레드를 실행하기 시작하고, 실행 시간(execution time)이 지난 후, 이 스레드를 대기하도록 하며, 만일 필요하다면 스레드의 데이터를 메모리 수단에 저장하고 새로운 스레드의 수행을 시작하는데, 새로운 스레드는 이전 스레드와는 동일하거나 또는 상이한 프로세스를 가질 수 있다. 스케줄러가 모든 스레드를 수행한 후에, 스케줄러는 새로운 실행 라운드(execution round)를 시작한다. 또는, 어떤 스레드의 실행 간격(execution interval)도 달성되지 않았다면, 소위 아이들 스레드(idle thread)가 수행된다. 그러나, 전술한 스레드들의 실행 시간 중에는, 인터럽트 리퀘스트가 발생할 수 있고, 이러한 인터럽트 리퀘스트들의 우선 순위에 기초하여 오퍼레이팅 시스템이 인터럽트 서비스 루틴을 수행하거나 스케줄러가 인터럽트 서비스를 수행할 적합한 시간을 결정하고 인터럽트가 발생한(interrupted) 스레드를 계속 수행한다. 예를 들면, 키패드 인터페

이스에 의해서 발생한 인터럽트 상황에서는, 키패드 인터럽트 프로그램이 입력된 키의 코드를 독출하고 그 코드를 임시 저장 장소에 저장한다. 인터럽트 서비스 루틴이 완료된 후, 오퍼레이팅 시스템의 스케줄러는 키 입력을 감지하는 프로그램이 동작하도록 실행 시간을 결정하고, 키패드 인터럽트에 대하여 결정된 우선 순위가 인터럽트가 발생한 프로그램 스레드의 우선 순위보다 앞서지 않으면 인터럽트가 발생한 프로그램 스레드로 복귀한다.

첨부된 도 1은 단일 오퍼레이팅 시스템의 계층 구조를 예시하는 도면이다. 오퍼레이팅 시스템의 최하위에는 커널이 위치하며, 그 상위 단계에는 메모리 콘트롤에 관련된 부분이 위치하고, 세 번째 레벨은 파일 시스템 부분을 구비한다. 이러한 세 레벨의 상위에는 예를 들어 코맨드 인터프리터, 프로세스(미도시), 인터페이스 프로그램, 및 스케줄러 펑션 등이 존재한다.

이하, 단일 오퍼레이팅 시스템의 커널 구조가 예시된다. 커널의 동작의 중요한 부분은 인터럽트 핸들링, 프로세스의 콘트롤 및 주변 장치들의 콘트롤이다. 인터럽트 핸들링에는 인터럽트가 발생한 프로그램의 데이터를 저장하는 동작 및 정확한 인터럽트 서비스 루틴이 실행되도록 콘트롤하는 동작 등이 있다. 프로세스의 콘트롤은 예를 들어 프로세스를 생성(creating)하고, 실행 시간을 결정하고(스케줄링하고), 프로세스의 동작을 종료하고, 타이밍을 맞추는 등의 작업 등에 해당한다. 주변 장치의 콘트롤 펑션(control function)에는, 예를 들어 데이터 전송을 시작하고 각 주변 장치에 관련된 하나 또는 그 이상의 인터럽트를 핸들링 하는 작업이 포함된다. 오퍼레이팅 시스템의 관점에서 보면, 프로세스는 다음 세 가지 모드 중 하나의 모드에서 이루어진다. 그 모드들은 준비 모드(ready mode), 실행 모드(run mode), 또는 대기 모드(wait mode)이다. 이렇게 모드를 구별하는 것이 첨부된 도 2에 설명되어 있다. 실행 모드는 그 순간에 수행되는 프로세스 들을 포함하며, 단일 프로세서를 구비하는 시스템은 실행 모드에서 최대로 한번에 한 프로세스만 가진다. 이러한 실행 모드에서 실행되기 위하여 대기하는 프로세스들은 준비 모드에 있다. 스케줄러의 작업은 준비 모드에 있는 프로세스 중 실행 모드에서 실행될 하나의 프로세스를 선택하는 것이다. 대기 모드는 키 입력 및 일부 자원의 재배치 등과 같은 일부 동작을 기다리는 프로세스 들을 포함한다.

스케줄러는 최상위의 우선 순위를 갖는 프로세스를 준비 모드로부터 실행 모드로 전환한다. 실행 모드에서, 일반적으로 프로세스는 자신의 요구에 의하여(by its own request) 대기 모드로 들어간다. 자신의 요구의 예를 들면 대기 단계를 포함하도록 구현된 오퍼레이팅 시스템 서비스를 요구하는 것이다. 이러한 대기 단계가 종료되면, 대기 모드로부터 준비 모드로 이동된다. 예를 들어, 스케줄러가 들어 인터럽트 핸들링을 종료하기 위하여 다른 실행될 프로세스를 변화시키는 상황에서도 프로세스는 또한 실행 모드에서 준비 모드로 되돌려질 수 있다.

커널은 일반적으로 다수의 동적 정보 구조(dynamic information structure)를 포함하는데, 동적 정보 구조는 전형적으로는 양방향 체인 리스트(typically two-way chain list)이며 그 구조들은 오퍼레이팅 시스템의 영역 내에 위치한 자유 메모리 영역으로부터 할당된다. 예를 들어, 각 프로세스의 데이터는 프로세스 콘트롤 블록(PCB, Process Control Block) 내에 유지되는 것이 바람직하다. 프로세스 요소(process element)들은 시스템 내의 프로세스의 양의 상한선을 설정하는 크기를 가지는 프로세스 테이블(process table) 내에 수집된다. 첨부된 도 3에 도시된 이러한 예시(example)는, 예를 들어 기초적인 요소(basic element) 내에 다음의 정보를 포함한다.

- 프로세스 상태(status): 대기, 실행 준비 또는 실행

- 프로세스의 명칭(name)

- 프로세스의 우선 순위(priority)

- 인터럽트가 발생한 프로세스의 정보(환경, environment) 저장(스택, stack)을 위한 상태(status)

- 할당된 자원들의 데이터: 예를 들어 메모리, 장치, 파일 오픈(open files) 등

- 프로세스 할당의 트래킹(tracking)에 관련된 필드(field), 및

- 인증(authorisation)에 관련된 필드

위와 같은 정보는 단순히 프로세스 요소를 구현하는 하나의 예시일 뿐이며, 실재의 실시예에서 이러한 정보들은 상당부분 변화될 수 있고 프로세스 요소 내에 저장된 데이터 역시 매우 상이할 수 있다는 것에 유의하여야 한다.

프로세스 심벌(process symbol)은 프로세스의 명칭을 의미하며, 예를 들어 시퀀스 번호(sequence number)와 같은 것이다. 이러한 프로세스의 명칭에 기초하여 오퍼레이팅 시스템은 상이한 프로세스들을 서로 구별한다. 프로세스가 하나 이상의 스레드를 구비할 경우에는, 프로세스 심벌, 또는 프로세스 요소의 다른 필드들은 이 스레드의 개수 정보를 역시 포함할 수 있다. 프로세스의 상태를 알려주기 위하여, 상태 필드(status field)가 사용된다. 그러므로, 프로세스의 상태가 변화하면 오퍼레이팅 시스템은 이 상태 필드의 값을 변화시킨다. 스케줄러는 프로세스의 실행 시퀀스를 결정하기 위하여 우선 순위 필드(priority field)의 값을 사용하는데, 이 필드에 포함된 수치가 클수록, 그 프로세스의 우선 순위는 높다. 그러므로, 프로세스들의 우선 순위 필드 값을 비교함으로써, 대기 모드에 있는 프로세스들 중에서 스케줄러는 가장 높은 우선 순위 값을 가지는 프로세스를 선택하여 다음에 수행시킨다.

이하, 새로운 프로세스를 생성(creation)하는 과정이 설명된다. 본 예시에서, 프로세스는 장치의 메모리 수단 내의 파일로 저장되는데 메모리 수단의 예를 들면 기입 가능한 대량 메모리(writeable mass memory) 등이다. 프로그램은 파일 명칭에 따라서 인식된다. 오퍼레이팅 시스템의 커널은 프로세스를 구동하기 위한 서비스 루틴을 구비하는데, 이 서비스 루틴에 파라미터로 주어지는 명칭은 이 파일의 명칭인 것이 바람직하다. 구동 서비스 루틴(activation service routine)은 프로세스 테이블에서 이 프로세스를 위한 프로세스 요소를 형성하고 메인 메모리의 일정 영역을 이 프로세스를 위해 할당한다. 이러한 프로그램 파일의 구성 요소들은 프로그램을 위해서 예약된 메모리 영역에서 독출되고, 프로세스 요소(process element)의 필드들은 적합한 값으로 초기화되며, 따라서 프로세스는 실행되기 위해 대기하도록 설정될 수 있다. 프로세스에 사용되는 심벌의 예를 들면 다음의 자유 프로세스 심벌(free process symbol)이 있다. 프로세스의 상태는 준비치(value ready)로 설정되는 것이 바람직하다. 오퍼레이팅 시스템의 스케줄러가 이러한 자신의 스케줄링 프로시져(scheduling procedure)에 일치하여 실행되도록 프로세스를 콘트롤한다.

프로세스가 스레드로서 실행되는 오퍼레이팅 시스템에서는, 하나의 구동 서비스 루틴이 하나 또는 수 개의 스레드를 프로세스로부터 생성하고 프로세스 테이블에 있는 스레드들을 위한 프로세스 요소들을 생성하도록 하는 방식으로 전술한 프로세스 생성 과정을 개별적으로 적응(adapt)시키는 것이 가능하다.

더 나아가, 오퍼레이팅 시스템은 주변 장치를 콘트롤하기 위한 실질적인 특징을 가지는데 그것이 소위 장치 드라이버(device driver)들이다. 장치 드라이버는 디스플레이, 키패드, 코덱(codec) 등과 같은 주변 장치를 콘트롤하기 위해 필요한 루틴들 및 정보 구조체(information structure)를 구비한다. 필요한 루틴들을 예를 들면 장치의 초기와 공정, 독출/기입 공정, 터미널의 특성을 설정하는 것과 같은 콘트롤 루틴 및 장치에 관련된 인터럽트들을 핸들링하는 루틴 등이 있다. 장치에 관련된 변수 및 정보 구조체는 예를 들어 소위 장치 기술자(device descriptor) 상에 취합될 수 있다. 이러한 타입의 정보를 예를 들면 다음과 같다:

- 장치를 향한 전송 리퀘스트(transfer request)들의 파라미터 블록들이 체인처럼(chained) 정렬된 작업 대기 행렬(work queue). 동시에 판독(value) 및 기입(writing)이 가능하면, 그 경우에는 두 개의 작업 대기 행렬이 있는 것이다. 파라미터 블록은 전송 리퀘스트(transfer request)의 파라미터들을 구비하는데, 이러한 파라미터에는 예를 들어 데이터 버퍼의 위치에 관련된 정보, 전송될 블록의 번호 등이 있다.

- 장치 드라이버 루틴들의 주소

- 전송 콘트롤에 요구되는 장치 변수 및 모드 데이터

- 장치 특유(device specific)의 파라미터들

어떤 시스템에서는 장치 드라이버들은 오퍼레이팅 시스템의 프로세스들이다. 프로세스는 우선 전송 리퀘스트 메시지를 기다리며 무한 루프(eternal loop) 내에서 동작한다. 전송 리퀘스트 메시지를 수신하면, 장치 드라이버 프로세스는 전송을 시작하고 인터럽트 메시지(interrupt message)를 기다린다. 그러면, 오퍼레이팅 시스템의 인터럽트 핸들러가 인터럽트 메시지를 장치 드라이버 프로세스로 송신하는데 기여한다. 실용적 장치들에서는, 장치 드라이버 프로세스가 전형적으로 높은 우선 순위를 가지며 오퍼레이팅 시스템의 커널은 실시간으로 동작한다.

기입 가능한 대량 메모리에서 독출할 때, 독출 동작은 예를 들면 다음과 같은 방식으로 이루어진다. 장치 테이블에서 우선 각 장치 기술자의 주소가 탐색된다. 탐색 작업 이후에, 독출 루틴의 주소가 장치 기술자로부터 패치(fetch)되고 요구된다. 독출 루틴은 전송 리퀘스트 파라미터들로부터 파라미터 블록을 형성한다. 만약 전송이 상기 장치에 의하여 현재 이루어지고 있으면, 다시 말해 장치가 다른 프로세스에 의해 점유되고 있으면, 파라미터 블록은 작업 대기 행렬의 맨 뒤에 연결되고 대기 작업이 이루어진다. 만일 장치가 자유로우면, 즉 작업 대기 행렬이 비어 있으면, 파라미터 블록은 작업 대기 행렬의

첫 번째로 연결된다. 결과적으로 블록의 수는 쉬트 서피스(sheet surface), 트랙(track) 및 섹터(sector)의 수까지 변화하며 전송 작업은 예를 들어 장치 레지스터를 변경함으로써 구동된다. 이 작업이 이루어진 후, 전송이 종료될 때까지 대기한다. 전송 작업이 종료되면 장치 인터럽트가 발생한다. 오퍼레이팅 시스템의 인터럽트 핸들링이 장치 테이블로부터 장치 번호에 따라 장치 기술자의 주소를 탐색하며, 이로부터 더 나아가 인터럽트 처리 루틴(interrupt treatment routine) 및 그 변화를 탐색하여 인터럽트 핸들러의 프로그램 코드(program code)를 수행한다. 인터럽트 핸들러는 아무런 오류도 발생하지 않았다는 것을 조사하는 것이 바람직하다. 성공적인 전송 작업의 결과, 파라미터 블록이 작업 대기 행렬로부터 삭제되고(chained off) 전송 작업 동안 대기하고 있었던 프로세스가 호출된다. 대기 중인 전송 리퀘스트가 작업 대기 행렬에 존재하는 경우에, 후속 전송 동작이 실질적으로 지연 없이 구동된다.

주변 장치의 타입에 기초하여, 이러한 과정은 어느 정도 전술한 바와 달라질 수 있다. 디스플레이 및 프린터와 같은 문자 인쇄 장치들에 의하여 작업 대기 행렬은 인쇄될 문자들의 대기 행렬로 용이하게 대체될 수 있다.

많은 오퍼레이팅 시스템은 사용중인 메시지 전송 메커니즘을 포함하는데, 이 메커니즘을 통하여 프로세스들이 서로 메시지를 전송할 수 있다. 메시지 대기 행렬을 사용할 경우에, 하나 또는 수 개의 메시지 대기 행렬이 프로세스에 관련되고, 그 프로세스에 전송될 메시지가 이 메시지 대기 행렬에 기입될 수 있다. 일반적으로 프로세스 간 통신에 버퍼링(buffering)이 관련되는데, 이 버퍼링을 통하면 메시지의 송신기는 수신기가 메시지를 수신할 때까지 대기할 필요가 없다. 그러므로, 메시지를 기입하는 단계(writing phase)에서, 정보(상태 변수)가 프로세스를 기다리고 있다고 알릴 수 있도록 설정되는데 프로세스는 그 메시지를 적합한 단계에서 독출한다.

본 발명의 바람직한 실시예에 의한 내장형 시스템에서는, 즉, 도 4a에 도시된 통신 장치(1)에는 두 개의 오퍼레이팅 시스템이 사용된다. 제 1 오퍼레이팅 시스템은 본 명세서에서는 OS_A라는 부재로 표시되는데, 주로 이동국 펑션(mobile station functions)을 구현하기 위하여 사용된다. 그리고, 제 2 오퍼레이팅 시스템은 본 명세서에서는 OS_B라는 부재로 표시되는데, 주로 데이터 프로세스 펑션(data process functions)을 구현하기 위하여 사용된다. 데이터 프로세스 펑션은 주로 개인용 컴퓨터 및 이와 같은 펑션들로부터 알려졌는데, 그 펑션들은 예를 들어 파일을 읽는 것, 기입 가능한 대량 메모리에 기입하는 것, 데이터를 프린트하는 것, 및 소위 구조화 펑션(organisation functions)(노트북, 콘택트 데이터의 유지 등) 등과 같은 응용 프로그램을 수행하는 것 등이 있다. 단일 프로세서(2)를 통신 장치(1)와 연결하여 사용하는 것이 바람직하기 때문에, 동시에 두 오퍼레이팅 시스템들(OS_A, OS_B)의 특징이 유지될 수 있도록 하는 방법으로 상이한 오퍼레이팅 장치들(OS_A, OS_B)의 펑션이 연결되는 것이 필요하다. 더 나아가, 필요할 경우 펑션들의 실재 시간이 하나의 척도(criterion)이다. 이제부터, 여러 실시예들에 비추어 어떻게 이렇게 상이한 오퍼레이팅 시스템들(OS_A, OS_B)이 동일한 프로세서(2)와 연결되어 결합(join)될 수 있느냐에 대해 설명된다.

도 5는 제 2 오퍼레이팅 시스템(OS_B) 아키텍쳐(architecture)의 한 예를 도시하는 도면이다. 각 블록은 특정 프로세스(certain process), 스레드, 접속 인터페이스 등을 예시한다. 블록들은 사용자 모드(USER) 또는 특권 모드(SVC, Supervisory mode) 중 어느 모드가 상기 블록 내에서 사용되느냐의 정보에 따라 더 나뉜다. 더 분리된 부분은 인터럽트 모드인데 도 5에서 블록(501)으로 도시되어 있다. 인터럽트 모드는 고속 인터럽트 모드(FIQ), 정규 인터럽트 모드(IRQ) 또는 소프트웨어 인터럽트 모드(SWI)를 사용한다. 제 2 오퍼레이팅 시스템(OS_B)의 커널(블록 502)은 특권 모드(SVC)에서 동작하도록 설정된다. 이에 반하여, 제 2 오퍼레이팅 시스템의 스케줄러(SCH_B, 블록 503), 접속 인터페이스(블록 504), 데이터 프로세서 및 이와 같은 프로세서의 달력 응용 프로그램(calendar application)과 같은 사용자 프로세스의 스레드들(506, 507 및 508) 뿐만 아니라 장치 드라이버의 접속 인터페이스(블록 505)는 사용자 모드(USER)에서 동작하도록 설정되는데, 이들은 예를 들어 오퍼레이팅 시스템의 스택(stack, 미도시) 또는 다른 중요한 메모리 블록(crucial memory block)들에 대한 접근권(access)을 가지고 있지 않다. 특권 모드는 또한 장치 드라이버(블록 509), 아이들 스레드(블록 510) 및 소위 존재 가능한 수퍼 스레드(super thread, 블록 511)들을 구비한다. 아이들 스레드(510)은 오퍼레이팅 시스템이 다른 프로세스가 실행되지 않을 경우에 실행하는 일정 종류의 대기 프로세스(waiting process)이다. 수퍼 스레드(511)란 본 명세서에서는, 정규 스레드(normal thread, 506, 507, 508)들의 응답 시간에 비하여 훨씬 작은 응답 시간을 갖는 스레드를 의미한다. 수퍼 스레드(511)의 응답 시간을 예로 들면 약 수백 마이크로초(microseconds) 일 수 있는데, 반면에 일반 스레드(506, 507, 508)들의 응답 시간은 수십 밀리초(milliseconds) 들이다. 본 발명은 또한 예를 들어 수퍼 스레드(511)를 구비하지 않는 오퍼레이팅 시스템과 관련해서도 동작할 수 있다.

오퍼레이팅 시스템의 커널(502)에는 가장 높은 우선 순위가 부여된다. 전술한 응답 시간 요구사항(requirement) 때문에, 수퍼 스레드(511)에는 일반 스레드(506, 507, 508)의 우선 순위보다 높은 우선 순위가 부여된다. 본 명세서에서 응답 시간이라고 하면, 인터럽트 리퀘스트의 도착 시점에서부터 스레드가 실행될 순간까지의 시간을 의미한다. 이러한 응답 시간은 기대치(expectation value)이므로, 현실적으로 일반 동작에서는 이러한 기대치보다 작거나 응답 시간이 최대일 때 기

대치와 같은 값을 갖는다. 과대 부하(excessive loading)가 발생하는 상황에서는, 응답 시간은 가끔 기대치를 초과할 수 있다. 응답 시간은 또한 인터럽트 서비스 루틴을 수행하는 동안 도과하는 시간을 포함한다. 실제로, 가급적 인터럽트 서비스 루틴을 단축시켜 인터럽트 서비스 루틴이 고속으로 동작하도록 구성하는 것이 목적이다.

본 발명의 바람직한 실시예에 의한 하나의 내장형 시스템은 첨부된 도 4a에서 통신 장치(1)에 의해 도시된다. 본 발명의 바람직한 실시예에 의한 하나의 내장형 시스템은 예를 들어 프로세서(2)를 구비하는데, 프로세서(2)는 예를 들어 마이크로프로세서 이거나 마이크로 콘트롤러의 일부이고, 도 4a에 도시된 블록도 중 적어도 일부의 블록이 마이크로 콘트롤러의 펑션 블록들로 구성될 수 있다. 프로세서(2)에 덧붙여서, 콘트롤 펑션 중 일부가 소위 응용 주문형 집적 회로(3, ASIC, Application Specific Integrated Circuit)를 이용하여 본 발명의 바람직한 실시예에서 실장된다. 개별 데이터 전송 버스(separate data transmission bus, 4)가 프로세서(2) 및 응용 주문형 집적 회로(3) 간에 실장될 수 있는데, 개별 데이터 전송 버스가 실장되면 데이터 전송 속도가 증가되고, 제 2 데이터 전송 버스(5)에는 부하가 걸리지 않는다. 이동국 펑션을 획득하기 위하여, 통신 장치에는 송신기/수신기 유니트(transmitter/receiver unit, 6), 송신/수신용 안테나(ANT), 디지털 신호 처리 유니트(DSP, Digital Signal Processing unit), 음향 신호를 코딩/디코딩하기 위한 코덱(codec, 8)이 제공되고, 이동국 펑션을 사용하기 위한 제 1 키패드(9) 및 정보를 사용자에게 표시하기 위한 제 1 디스플레이 장치(10)가 제공된다. 또한, 통신 장치(1)는 음성 블록(11, audio block)을 구비하는데 이 음성 블록(11)에서는 마이크로폰(12)에서 형성된 아날로그 신호를 변환하는데 필요한 아날로그/디지털 변환이 수행되는 것 뿐만 아니라, 라우드 스피커(loud speaker, 13)로 보내지는 신호를 위한 디지털/아날로그 변환이 수행된다. 데이터 프로세싱 기능을 수행하기 위하여, 통신 장치(1)에는 제 2 키패드(14), 제 2 디스플레이(15) 및 예를 들어 프린터를 통신 장치(1)에 결합하여 연결시키는 인터페이스 블록(16)들이 제공된다. 키패드(9, 14) 및 디스플레이(10, 15)를 위하여 일반적으로 사용되는 통상적 이름은 사용자 인터페이스(UI, User Interface)인데, 사용자 인터페이스(UI)를 통하여 사용자 및 통신 장치(1) 간의 통신이 적어도 일부분 수행된다. 어떠한 응용 프로그램에서는, 사용자 인터페이스(UI)에는 마이크로폰(12) 및 라우드 스피커(13)와 같은 음성 수단(audio means)이 포함될 수 있다. 도 1에 따른 통신 장치(1)는 두 개의 사용자 인터페이스(UI1, UI2)를 예시하는데, 제 1 사용자 인터페이스(UI1)는 제 1 키패드(9) 및 제 1 디스플레이(10)를 구비하고, 제 2 사용자 인터페이스(UI2)는 제 2 키패드(14) 및 제 2 디스플레이(15)를 구비한다. 제 1 사용자 인터페이스(UI1)는 일반적으로 이동국 동작에 관여하고 제 2 사용자 인터페이스(UI2)는 일반적으로 데이터 프로세싱 동작에 관여하지만, 두 사용자 인터페이스(UI1, UI2) 모두 필요한 경우 이동국 동작 및 데이터 프로세싱 동작과 관련되어 사용될 수 있다. 사용자 인터페이스(UI1, UI2)들은 반드시 분리되어야 하는 것은 아니며, 예를 들어 단일 디스플레이 및/또는 단일 키패드를 구비하는 단일 사용자 인터페이스로 실장될 수 있다.

사용자 인터페이스(UI1, UI2)들은 소위 말하는 터치 스크린 방식이 사용되는 형태로 실장될 수도 있다. 이러한 방식에서는, 접촉감응 수단(touch sensitive means)이 디스플레이 상부(top)에 위치하는 것이 바람직하며, 이 수단이 예를 들어 사용자의 손가락의 터치에 반응한다. 접촉감응 수단들의 동작 원리는 잘 알려져 있으며, 예를 들면 커패시턴스 변화 또는 저항치 변화에 기초하고 있다.

통신 장치(1)는 더 나아가 메모리(17)를 구비하는데, 메모리(17)는 예를 들어 부팅 코드(booting code)를 저장하는 독출 전용 메모리(ROM), 실행될 응용 프로그램을 탑재하고 사용 중 필요한 데이터 및 기입 가능한 대량 메모리로부터의 데이터를 저장하기 위한 임의 접근 메모리(RAM), 그리고 플래시 메모리(FLASH memory) 및/또는 비 휘발성 임의 접근 메모리(NVRAM, Non-Volatile Random Access Memory)등을 구비하는 것이 바람직하다. 통신 장치(1)의 펑션 블록들은 제 2 데이터 전송 버스(5)에 의하여 연결되어 있는데, 제 2 데이터 전송 버스(5)는 예를 들면 어드레스 버스, 데이터 버스, 및 콘트롤 버스를 구비한다. 그러나, 명확성을 기하기 위하여, 이러한 버스들은 당업계의 기술자들에게는 알려져 있는 것이기 때문에 분리되어 도시되지 않았다. 파워 서플라이(power supply)는 배터리(18)로 구성되고 전압 형성 블록(voltage forming block, 19)에 연결되는데, 전압 형성 블록(19)은 전압 안정기(voltage regulator), 전압 변환기(voltage converter) 또는 이와 같은 장비를 가능한 위치에 구비하여 필요한 상이한 동작 전압들(Vcc1, Vcc2)을 생성한다. 전압 형성 블록은 더 나아가 배터리(18)의 전하량을 통신 장치(1)가 동작을 멈출 정도의 낮은 레벨로 낮추기 위한 신호를 형성한다. 그러므로, 배터리(18)는 재충전되어야 하며, 그렇지 않으면 동작 전압은 턴오프 된다. 실제 실시예에서는, 전하량 상태에 관한 정보가 인터럽트 리퀘스트를 프로세서(2)로 발생시키고 이러한 인터럽트 서비스 루틴과 결합되어 데이터 저장 요구치(data storage requirement)가 설정되고, 그러면 오퍼레이팅 시스템들(OS_A, OS_B)이 디스플레이들(10, 15)들로 하여금 어큐뮬레이터(accumulator, 18)를 비우고 데이터를 메모리(17)에 저장하기 시작하라는 메시지를 생성한다. 그러면, 동작 전압을 턴오프 하기 위한 콘트롤은 저절로 수행된다.

통신 장치(1)는 또한 예를 들어 주변 장치 중 일부에 다른 프로세서들을 구비할 수 있지만 이것은 본 발명에는 중요한 사항이 아니며, 그 이유는 다른 프로세서들이 구비되는 것은 오퍼레이팅 시스템 펑션을 수행하기 위한 것과는 다른 목적을 위한 것이기 때문이다.

예시된 실시예에서 사용된 프로세서(2)는 ARM7 시리즈에 속하는 프로세서로서, 어드밴스트 RISC 머신스(Advanced RISC Machines) 사에서 제조된 것이다. 이 프로세서는 소위 축소 명령 세트 컴퓨터 형태의 프로세서이다. 그러나, 본 발명은 ARM7 타입의 프로세서 또는 축소 명령 세트 컴퓨터 타입의 프로세서에만 한정되는 것은 아니며, 본 발명은 다른 타입의 프로세서와 결합되어 사용될 수도 있다. 첨부된 도 4B는 ARM7 시리즈 타입의 프로세서 중 하나의 간략화된 블록도이다. 프로세서(2)의 동작과 관련하여, 중심 블록은 코맨드 인터프리터/콘트롤 블록(command interpreter and control block, 401)이다. 이 코맨드 인터프리터/콘트롤 블록(401)의 목적은 예를 들면 프로그램 코맨드를 해석하고 코맨드에 의하여 요구되는 수단(measure)들을 콘트롤함으로써, 고속 인터럽트 라인(nFIQ) 및 일반 인터럽트 라인(nIRQ)을 통해 도달하는 인터럽트 리퀘스트에 반응하고, 소프트웨어 인터럽트 리퀘스트에 반응하고, 외부 클럭 신호(미도시)의 타이밍 신호 및 이와 같이 프로세서의 동작을 위한 신호의 형태에 반응하고, 예를 들어 데이터를 독출하고 메모리(17) 및 프로세서(2) 간에 데이터를 기입하기 위한 콘트롤 신호를 생성하는 것 뿐만 아니라 그 순간에 어드레스 버스(402)에 필요한 어드레스 데이터의 형식(formation)를 콘트롤 하는 것 등이다.

데이터 버스의 독출 레지스터(reading register, 403)는 데이터 버스(404)로부터 전달되는 프로그램 코맨드를 데이터 버스의 독출 레지스터(403)의 내부 코맨드 대기 행렬(command queue, 미도시)로 위치시키고, 데이터를 데이터 레지스터(미도시)로 전송한다. 코맨드 인터프리터/콘트롤 블록(401)은 내부 코맨드 대기 행렬에서 다음에 수행될 프로그램 코맨드를 독출하고 독출한 코맨드를 해석한다. 필요하다면, 코맨드 인터프리터/콘트롤 블록(401)은 데이터를 데이터 레지스터로부터 독출하고 독출한 데이터를 제 2 내부 데이터 버스(405, B 버스)로 전송하는데, 제 2 내부 데이터 버스(405, B 버스)는 인가 받은 정보를 배럴 시프터(barrier shifter, 406)를 통하여 제 1 내부 데이터 버스(409, A 버스)가 연결되어 있는 연산 논리 장치(ALU, Arithmetic Logic Unit)로 전송한다. 같은 방법으로, 코맨드 인터프리터/콘트롤 블록(401)은 데이터를 데이터 버스의 기입 레지스터(writing register, 408)에 기입하는 작업을 콘트롤하고, 기입 레지스터(408)로부터 통하여 데이터는 데이터 버스(404)로 전송된다.

논리 연산 장치(ALU, Arithmetic Logic Unit, 407)에서는 예를 들어 가산 및 감산 뿐만 아니라 논리 연산이 알려진 방법으로 수행된다. 논리 연산 장치(407)로부터 발생된 데이터는 제 3 내부 데이터 버스(410, ALU 버스)를 따라서 어드레스 레지스터(411) 뿐만 아니라 내부 데이터 레지스터들(레지스터 뱅크)을 포함하는 데이터 레지스터 블록(412)으로 전송된다. 데이터 레지스터 블록(412)은 데이터를 저장하기 위해 마련된 레지스터, 상태 레지스터(status register) 및 예를 들어 통신 장치의 메모리(17)와 같은 프로그램 메모리 영역으로부터 프로그램 코맨드들을 보여주기 위한 프로그램 카운터(PC, Program Counter)등을 구비한다. 이러한 목적을 위하여, 내부 어드레스 버스(413)가 데이터 레지스터 블록(412)으로부터 어드레스 레지스터(411)까지 연결한다.

어드레스 레지스터(411)는 어드레스 카운터 블록(414)에 연결되는데, 어드레스 카운터 블록(414)에서 프로그램 카운터의 값은 일반적으로 하나 증가되어, 프로그램 코드 내에서 다음에 수행될 프로그램 코맨드가 일반적으로 위치하는 프로그램 코드의 다음 위치를 가리키게 된다. 어떠한 상황에서는, 서브-프로그램 리퀘스트(sub-program request)와 같은,. 프로그램 코맨드 또는 인터럽트 리퀘스트가 프로그램 코드 상의 다른 위치로의 전송을 야기할 수 있는데, 그러면 코맨드 인터프리터/콘트롤 블록이 이 어드레스를 프로그램 카운터에 설정하고 이 어드레스가 어드레스 레지스터(411)의 값이 되도록 전달한다. 어드레스 변화 버스(address changing bus, 415)가 어드레스 카운터 블록(414)으로부터 어드레스 레지스터(411) 및 데이터 레지스터 블록(412)을 향해 연결된다.

프로세서(2)는 또한 곱셈 및 나눗셈을 수행하기 위한 곱셈 블록(multiplication block, 416)을 구비한다. 곱셈 블록(416)은 제 1 내부 데이터 버스(409) 및 제 2 내부 데이터 버스(405)에 연결된다. 전술된 프로세서(2)는 소개된 다른 접속 라인들(connection lines) 외의 다른 접속 라인들을 구비한다. 그러나, 본 명세서에서 이들을 자세히 설명할 필요는 없다.

프로세서(2)의 데이터 레지스터 블록(412) 내의 데이터 레지스터의 일부는 프로세서의 모든 동작 모드에서 사용될 수 있다. 더 나아가, 상이한 동작 모드에서 이러한 데이터 레지스터들은 예약(reserve)되지만 다른 동작 모드에서는 차단(block)된다. 이러한 실시예(arrangement)에서, 인터럽트 상황(interrupt situation)에 결부하여 특정 데이터를 저장해야 할 필요성을 줄이는 것이 가능하다. 덧붙이면, 상이한 동작 모드에서 특정한 스택 포인터(stack pointer)가 제공되는데, 각 동작 모드는 필요하면 그 스택을 위해 예약된 메모리(17)의 고유의 메모리 영역을 가지는 것이 바람직하다.

프로세서(2)는 적어도 세 개의 동작 모드에 설정되는 것이 바람직하다. 세 가지 모드는 자원의 사용이 제한되는 사용자 모드, 및 프로세서의 모든 자원이 사용될 수 있는 특권 모드(관리자 모드) 또는 프로세서(2)가 프로그램 코드에서 식별되지 않은 코맨드를 검출했을 경우에 일반적으로 사용되는 특권 모드의 특별 모드인 미결정 모드(undefined mode)이다. 바람직한 실시예에서, 이러한 미결정 모드는 본 명세서에서 자세히 후술된 하나의 모드처럼 의도적으로 사용된다. 프로세서(2) 내에서, 미결정 모드는 다음의 목적에서 유용하다. 필요하다면, 다른 특권 모드가 미결정 모드에 상응하는 모드로 사용될

수 있다. 더 나아가, 프로세서(2)에서 세 가지 타입의 인터럽트가 사용될 수 있다. 그 세 가지 인터럽트는 고속 인터럽트(FIQ, Fast Interrupt Request), 정규 인터럽트 리퀘스트(IRQ, normal Interrupt Request) 및 프로그램된 인터럽트(SWI, Software Interrupt Request)들이다. 고속 인터럽트(FIQ)는 정규 인터럽트 리퀘스트(IRQ)에 비하여 높은 레벨의 우선 순위를 가지는데 정규 인터럽트 리퀘스트(IRQ)는 또한 프로그램된 인터럽트(SWI) 보다는 높은 우선 순위 레벨을 가진다. 그러므로, 응답 시간의 관점에서 보면, 중요 인터럽트(critical interrupt)는 고속 인터럽트(FIQ)를 이용하여 구성되는 것이 바람직하다.

통신 장치(1)를 구동시킬 때, 예를 들어 메모리 검사를 수행하는 등의 프로그램 코맨드들이 구동 프로그램 코드(activation program code) 내에 형성되고, 구동 프로그램 코드는 바람직하게는 프로세서(2)의 콘트롤 하에 통신 장치의 메모리(17)에 저장되거나, 독출 전용 메모리 또는 비 휘발성 임의 접근 메모리 내에 저장된다. 구동을 시작하는 것과 관련하여, 파일의 콘트롤 루틴(control routine)들 역시 탑재되는데, 그러면 오퍼레이팅 시스템들(OS_A, OS_B) 의 프로그램 코드들이 메모리(17)에 탑재되는 작업은 필요에 따라 파일의 콘트롤 루틴에 의해 수행된다. 이것은 많은 데이터 프로세싱 장치 및 오퍼레이팅 시스템에서 잘 알려진 기술이다. 오퍼레이팅 시스템들(OS_A, OS_B)의 프로그램 코드들은 또한 예를 들어 판독 전용 메모리(ROM), 비 휘발성 메모리(NVRAM) 또는 전기적 소거 가능한 피롬(EEPROM, electrically erasable programmable ROM) 내에 저장될 수 있다. 그러므로, 오퍼레이팅 시스템들(OS_A, OS_B)을 직접 저장 위치(storage location)에서 사용하는 것이 가능하다. 이것을 적소 실행(XIP, Execute In Place) 기술이라 한다.

오퍼레이팅 시스템들(OS_A, OS_B)을 탑재한 후에, 오퍼레이팅 시스템들(OS_A, OS_B)이 실행된다. 그러면, 첫 번째 단계에서는 예를 들어 제 2 오퍼레이팅 시스템(OS_B)의 시동 동작(starting operation)을 수행하는 것인데, 그 과정에서 다른 여러 가지 중에서 프로세스 테이블, 메모리 영역, 인터럽트 서비스 루틴, 메시지 대기 행렬(message queue) 및 오퍼레이팅 시스템에 상응하는 다른 기술자(descriptor)가 설정되고, 장치 드라이버 및 데이터 타입들이 탑재되며 인터럽트가 허용된다. 다음 단계에서, 제 1 오퍼레이팅 시스템(OS_A)의 상응하는 시동 동작의 수행이 시작된다. 제 2 오퍼레이팅 시스템(OS_B)의 시동 동작이 우선 실행된 후에, 예를 들어 오퍼레이팅 시스템(OS_A)의 인터럽트 펑션(interrupt function)과 같은 제 1 오퍼레이팅 시스템(OS_A)의 시동 동작이 수행되는 동안 인터럽트의 시작 및 인터럽트의 종료는 동적으로 인터럽트 핸들러에 연결된다(예를 들어, 펑션의 시작 인터페이스가 메모리(17) 내에 설정된다). 첨가하여 설명하면, 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 스케줄링하기 위하여 인터럽트 핸들러에 인터페이스가 형성된다.

무엇보다도, 장치 드라이버는 주변 장치의 초기 모드 세팅 작업을 수행한다. 고정 영역(fixed area) 즉, 정적 할당 영역(static allocation)이 데이터 메모리(RAM) 내에 예약되어, 정규 동작 도중에 필요에 따라 메모리 영역이 할당되는(동적 할당) 방식에 비하여 충분한 동작 시간을 더욱 잘 보장할 수 있도록 하는 것이 바람직하다.

예를 들어 아이들 스레드(idle threads)를 실행하는 것과 같은 시동 동작에 이어서, 어떤 응용 프로그램을 탑재하고 실행하는 것이 가능하다. 예를 들면, 이동국 펑션을 시작하는 것이 가능한데, 만약 다음의 펑션들이 통신 장치(1)에 설정되어 있다면 통신 장치(1)는 전화, 메시지, 팩스 등을 수신할 준비가 되고, 사용자는 그 또는 그녀가 원한다면 전화를 걸 수 있고, 메시지를 보내거나 팩스를 보낼 수 있다.

본 발명의 통신 장치(1) 내에 인터럽트가 발생할 경우에는, 프로세서(2)가 인터럽트 인터프리터(interrupt interpreter) 또는 인터럽트가 발생된 스레드의 상태를 저장하는 디스패처(dispatcher)를 수행한다. 그 다음 단계로서, 디스패처가 인터럽트 서비스 루틴을 시작한다. 어떤 인터럽트 서비스 루틴이 시작되느냐는 전형적으로 인터럽트 리퀘스트의 원인(cause)에 의존한다. 예를 들어, 프로세서(2)가 상태 레지스터를 가지고 있는 경우에 이 상태 레지스터의 내용이 인터럽트의 원인에 대한 정보를 제공한다는 것이 결정된다. 예를 들어, 키패드 인터럽트의 결과, 그에 상응하는 상태 레지스터는 상이한 논리 상태(예를 들어 논리 0 상태에서 논리 1 상태로)로 변화하고, 프로세서(2)가 이 레지스터의 상이한 비트를 검사(examine)하고 그에 상응하는 인터럽트 서비스 루틴을 시작한다. 수 개의 인터럽트 리퀘스트가 유효할 경우에, 높은 우선 순위를 가지는 인터럽트 서비스 루틴이 우선 시작된다.

또한, 처리 가능한 수 개의 인터럽트 라인들(nFIQ, nIRQ)을 가지는 프로세서(2)가 알려져 있는데, 인터럽트 라인(nFIQ, nIRQ)의 상태 변화가 인터럽트 리퀘스트를 수행한다. 특정한 인터럽트 서비스 루틴이 각 인터럽트 라인(nFIQ, nIRQ)을 위해서 결정될 수 있다. 더 나아가, 이러한 인터럽트 서비스 루틴의 어드레스들은 테이블 형태를 가질 수 있고, 그 테이블 안에서 프로세서(2)가 인터럽트 서비스 루틴의 시작 어드레스를 인터럽트에 따라 패치(fetch)하고 이 결과를 자신의 프로그램 카운터의 값으로 설정하는데, 다음에 실행될 코맨드가 이 어드레스에 존재한다.

본 발명에서, 수 개의 인터럽트 서비스 루틴들 또는 심지어 인터럽트들(FIQ, IRQ, SWI)에 결정된 모든 인터럽트 서비스 루틴을 변화하는 것 역시 가능하다. 예를 들면, 어떤 응용 프로그램은 인터럽트들(FIQ, IRQ, SWI)과 관련되어 사용될 수 있는 새로운 인터럽트를 결정할 수 있다. 그러한 응용 프로그램이 실행되면, 프로세서(2)는 메모리(17) 내의 인터럽트 서비

스 루틴을 변형한다. 본 발명의 바람직한 실시예들에 따르면, 프로세서(2)가 다른 인터럽트 서비스 루틴을 사용해야만 하는 이러한 인터럽트의 시작 어드레스만 변경하는 것 역시 가능하다. 인터럽트 서비스 루틴의 실제 코드(actual code)는 메모리(17) 내에 저장된다. 이것은 인터럽트 서비스 루틴의 시작 어드레스들로 구성된 어떤 종류의 어드레스 테이블이 존재한다는 것을 의미한다. 또한, 앞부분에 제시된 바와는 다르게, 어떤 인터럽트 서비스 루틴이 인터럽트들(FIQ, IRQ, SWI)을 위해 사용되어야 할지를 정의하는 다른 방법들이 존재할 수도 있다.

예를 들면 키패드 버퍼를 독출하고 데이터를 메모리에 저장하는 등과 같이 인터럽트 서비스 루틴 내에 요구되는 필요 동작들이 수행될 수 있다. 뿐만 아니라, 인터럽트 서비스는 일부 스레드 또는 오퍼레이팅 시스템에 전송되는 신호를 생성할 수 있다. 본 발명의 바람직한 실시예에 따른 통신 장치(1)의 동작에 대한 더 자세한 설명은 본 명세서에서 후술될 것이다.

인터럽트 서비스의 응답 시간은 전형적으로는 전술된 수퍼 스레드(511)의 예를 들어 약 100 마이크로초에 해당하는 응답 시간보다 훨씬 작다. 이러한 인터럽트 서비스의 응답 시간에 영향을 주는 요인들을 예를 들면 인터럽트가 프로그램 코드 내의 어떤 부분에서 얼마나 오랫동안 비활성화 되느냐의 사실 등이 있다. 인터럽트 취소(interrupt cancel)는 모든 인터럽트들 또는 특정 레벨보다 낮은 우선 순위 레벨을 가지는 인터럽트들에 관련될 수 있고, 또는 인터럽트들은 인터럽트들의 마스크 레지스터(mask register) 또는 이와 유사한 레지스터에 의하여 임시로(temporarily) 취소될 수도 있다. 그러므로, 인터럽트 리퀘스트는 메모리에 남게 되고, 인터럽트 서비스 루틴은 이 인터럽트 리퀘스트를 위하여 인터럽트 취소 동작이 제거되고 더 높은 우선 순위 레벨을 가지는 스레드가 수행되지 않는 단계에서 실행된다.

제 1 오퍼레이팅 시스템(OS_A)에서 제 2 오퍼레이팅 시스템(OS_B)으로 전환하는 것은, 제 1 오퍼레이팅 시스템(OS_A)의 스레드들 중 어느 것도 실행되지 않을 경우에 일어날 수 있다. 그러므로, 제 1 오퍼레이팅 시스템(OS_A)의 관점에서 실행 동작은 아이들 스레드(idle thread) 내에서 발생한다.

통신 장치(1)가 동작하는 동안, 상이한 오퍼레이팅 시스템들(OS_A, OS_B) 내에서 동작하는 응용 프로그램 간에 정보를 전송할 필요성도 존재할 수 있다. 이것은, 두 오퍼레이팅 시스템들(OS_A, OS_B) 모두가 적어도 일부분씩 공통적인 자원을 공유하는 경우에 특히 필요하다. 예를 들면, 키패드(9, 14)는 공통적인 키(common key)를 구비할 수 있거나, 분리된 키패드(9, 14) 대신에 공통 키패드가 사용될 수 있다. 반면에, 이러한 자원들의 콘트롤에 단일 장치 드라이버를 가지고 기여하는 것이 가끔 바람직한데, 그런 경우에는 접속 인터페이스들은 상이한 오퍼레이팅 시스템들(OS_A, OS_B)로부터 구현되어 있다. 그러면, 하나 이상의 오퍼레이팅 시스템들(OS_A, OS_B)이 동일한 자원을 동시에 사용하려 시도할 때, 그러한 상황을 콘트롤 해야하는 것이 용이하게 된다.

이제부터 본 발명에 의한 통신 장치의 동작이, 상이한 오퍼레이팅 시스템들(OS_A, OS_B)의 관점에서 설명된다. 본 발명의 바람직한 실시예에서, 제 1 오퍼레이팅 시스템(OS_A)은 소위 말하는 실시간 오퍼레이팅 시스템이고, 특정한 실행 시간 요구사항은 이 오퍼레이팅 시스템에 맞추어 결정되어 있다. 제 2 오퍼레이팅 시스템(OS_B)은 실행 시간에 관하여 중요한 것은 아니지만, 사용자에게 편의를 제공해야 한다는 측면에서, 제 2 오퍼레이팅 시스템(OS_B) 역시 바람직한 시간 내에 요구된 동작을 수행해야 한다.

제 1 오퍼레이팅 시스템(OS_A)의 스레드들은 그들을 위해 결정된 우선 순위를 가지고 있는데, 우선 순위는 예를 들면 내장형 시스템의 설계 단계(design phase)에서 결정된다. 우선 순위를 사용함으로 인해, 예를 들어 실행 순서 및 중요한 스레드들의 반응 시간들에 영향을 끼치는 것이 가능해진다. 제 1 오퍼레이팅 시스템의 스케줄러(SCH_A)는 실행 과정(execution round)이 시작될 때 가장 먼저 실행되어야 할 스레드에 가장 높은 우선 순위를 설정한다. 실제 시스템에서, 이것은 본 발명의 바람직한 실시예에 의한 통신 장치(1)의 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A)의 스케줄링 펑션(scheduling function)을 수행한다는 것을 의미하고, 스케줄링 펑션은 프로세서(2)의 프로그램 코맨드들에 의하여 실장된다. 스레드를 실행할 때, 프로세서(2)는 스레드 내에 프로그램된 프로그램 코맨드들에 따라서 조치(measure)를 수행한다.

제 1 오퍼레이팅 시스템(OS_A)에서 동작하는 프로세스의 모든 스레드들(THA1, THA2)을 실행한 후, 프로세서(2)는 제 2 오퍼레이팅 시스템(OS_B)의 동작을 수행한다. 예를 들면, 지연된 펑션(delayed function)이 지연된 서비스 루틴(DSR, Delayed Service Routine) 또는 대기하고 있는 스레드들(THB1, THB2)을 호출한다. 제 1 오퍼레이팅 시스템(OS_A)의 스레드들(THA1, THA2)은 인터럽트 핸들링이 일어나는 동안의 짧은 정지(short breaks) 외에는 언제나 동작하도록 허가(authorised)되었다. 이것에 대한 설명은 본 명세서에서 후술된다. 본 발명의 바람직한 이 실시예에서, 제 2 오퍼레이팅 시스템(OS_B)을 실행하는 것은 제 1 오퍼레이팅 시스템(OS_A)의 관점에서 제 1 오퍼레이팅 시스템(OS_A)의 첫 번째 스레드를 실행하는 것에 해당하는데, 이 첫 번째 스레드는 아이들 스레드(THA_IDLE) 또는 수퍼 스레드와 같은 것이다. 이 동작을 수행하기 위하여, 비록 제 1 오퍼레이팅 시스템(OS_A)이 본 실시예에 실장된 적당한 아이들 스레드를 구비하고 있지는 않지만, 제 1 오퍼레이팅 시스템(OS_A)에는 아이들 스레드(THA_IDLE)를 나타내는 정보 구조체(미도시)가 제공된다.

제 2 오퍼레이팅 시스템(OS_B)을 수행하기 위하여 이동(move)한 후, 프로세서(2)는 제 2 오퍼레이팅 시스템(OS_B)의 실행 스레드들(THB1, THB2)의 우선 순위 및 스케줄링 실무(scheduling practises)에 기초하여 제 2 오퍼레이팅 시스템(OS_B)의 스케줄링 평션을 수행한다. 우선, 지연된 서비스 루틴들(DSR) 중 일부가 대기중이면 지연된 서비스 루틴들(DSR)이 실행되고, 그 후 스레드들(THB1, THB2)의 수행이 시작되는데, 스레드들(THB1, THB2)의 수행 작업은 우선 순위에 따르는 것이 바람직하다. 실행되어야 했던 스레드들(THB1, THB2)을 수행한 후, 스레드들(THB1, THB2) 모두 실행 모드(run mode) 에 있지 않으면, 프로세서(2)는 제 2 오퍼레이팅 시스템(OS_B)의 아이들 스레드(THB_IDLE)로 이동하여 새로운 실행 과정(execution round)이 수행될 필요가 있을 때까지 아이들 스레드(THB_IDLE)에 머물게 된다.

제 2 오퍼레이팅 시스템(OS_B)을 수행하는 동안, 제 1 오퍼레이팅 시스템(OS_A)의 변화는 거의 아무 단계에서나 발생할 수 있다. 이러한 변화는, 인터럽트 리퀘스트 때문에 프로세서(2)가 이동하여 인터럽트 서비스 루틴을 수행하고, 제 1 오퍼레이팅 시스템(OS_A)은 후술하는 방식으로 구동되는 방식으로 이루어지는 것이 바람직하다. 제 2 오퍼레이팅 시스템(OS_B)으로 돌아가는 것은 제 1 오퍼레이팅 시스템(OS_A)이 아이들 스레드(THA_IDLE)보다 우선적으로 수행되는 경우에 이루어진다. 그러므로, 제 1 오퍼레이팅 시스템(OS_A)이 수행되는 것은 제 2 오퍼레이팅 시스템(OS_B)의 관점에서 보면, 인터럽트 서비스 루틴을 수행하는 것이다.

본 발명의 목적은 오퍼레이팅 시스템들(OS_A, OS_B)을 서로 상이한 오퍼레이팅 시스템들(OS_A, OS_B)의 프로그램 코드에 이루어지는 변화가 가급적 적은 방향으로 이루어지도록 두 오퍼레이팅 시스템들(OS_A, OS_B)을 연결하여 실장하는 것이다. 다시 말하면, 오퍼레이팅 시스템들(OS_A, OS_B)의 특성들이 응용 프로그램 개발자들의 관점에서 볼 때 너무 큰 정도로 변화하지 않도록 하는 방식으로 오퍼레이팅 시스템들(OS_A, OS_B) 연결 실장하는 것이 목적이다. 본 발명의 바람직한 실시예에 따른 통신 장치에서, 이러한 연결(connection)은 인터럽트 서비스 루틴 레벨 상에 이루어진다. 장치 인터럽트(FIQ, IRQ) 또는 프로그램 인터럽트(SWI)가 발생하면, 프로세서(2)는 이동하여 해당하는 인터럽트 서비스 루틴을 수행한다. 이 프로그램은 모든 인터럽트에 공통일 수도 있고, 또는 서로 상이한 장치 인터럽트(FIQ, IRQ) 또는 프로그램 인터럽트(SWI)를 위한 수 개의 상이한 인터럽트 서비스 루틴이 있을 수도 있다. 공통일 경우에, 프로세서(2)의 코맨드 인터프리터/콘트롤 블록(40)이 레지스터 등을 연구하여 필요한 경우 인터럽트의 원인을 결정한다. 인터럽트 서비스 루틴이 시작될 때, 인터럽트 시작 루틴에 첨가된 프로그램 호출(program call) 또는 평션 호출(function call)이 있는데, 이 루틴 내에서 오퍼레이팅 시스템들(OS_A, OS_B)의 내부 상태가 변화되고, 오퍼레이팅 시스템들(OS_A, OS_B)의 데이터가 인터럽트 서비스가 실행중임을 알린다. 인터럽트 서비스 루틴으로 전송하는 것, 서브 프로그램을 실행하는 것, 상태 데이터를 연구하는 것 뿐만 아니라 다른 상응하는 동작들은 모두 코맨드 인터프리터/콘트롤 블록(401)의 콘트롤 하에서 수행되는데, 그 기술은 본 명세서에서 전술된 바가 있고, 다시 말하면 당업자들에게는 공지된 기술이다.

계속하여, 인터럽트 핸들러의 펑션들이 수행되는데, 그것은 코맨드 인터프리터/콘트롤 블록(401)이 프로세서(2)에 전달된 인터럽트에 맞도록 결정된 프로그램 코맨드를 수행한다는 것을 의미하며, 이러한 프로그램 코맨드들은 메모리(17) 내에 저장되는 것이 바람직하다. 인터럽트 핸들러 내에서는, 예를 들어 무엇이 인터럽트를 야기했는지에 대하여 연구되고, 인터럽트 핸들링 작업을 수행하려면 제 1 오퍼레이팅 시스템(OS_A) 또는 제 2 오퍼레이팅 시스템(OS_B)의 스레드들 중 어느 스레드를 실행해야 하는지가 결정된다. 만약 개별 인터럽트 핸들링이 제 1 오퍼레이팅 시스템(OS_A)을 실행하도록 요구한다면, 인터럽트 핸들러는 이러한 정보를 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)에 설정한다. 만약 개별 인터럽트 핸들링이 제 2 오퍼레이팅 시스템(OS_B)을 실행하도록 요구한다면, 인터럽트 핸들러는 제 1 오퍼레이팅 시스템(OS_A)의 스레드들이 실행된 후에 수행될 지연된 서비스 루틴들(DSR)을 생성한다. 이러한 지연된 서비스 루틴들(DSR)은 제 1 오퍼레이팅 시스템(OS_A)의 수행중인 스레드들(running thread)을 통해서도 생성될 수 있다. 이 지연된 서비스 루틴들(DSR)은 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)에게 인터럽트 핸들링이 제 2 오퍼레이팅 시스템(OS_B) 내의 특정 스레드가 수행되기를 요구한다는 것을 알려준다. 어떠한 상황에서는, 인터럽트 핸들러가 이 데이터를 지연된 서비스 루틴들(DSR)을 사용하지 않고 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)에 설정하는 것이 역시 가능하다.

인터럽트 서비스 루틴 내에서는, 데이터 독출하여 버퍼에 저장하는 것과 같은 다른 동작 역시 가능한데, 그 버퍼 내에서는 그 데이터가 어드레싱하는 응용 프로그램의 스레드가 실행 단계(execution phase)에 있을 때 저장된 데이터가 독출된다.

인터럽트 프로그램이 종료될 때, 인터럽트 종료 펑션 리퀘스트가 추가되는데, 예를 들어 인터럽트 서비스 루틴의 실행이 종료됐음을 알리는 정보가 그 인터럽트 종료 펑션 리퀘스트에 설정된다. 더 나아가, 인터럽트 종료 펑션이, 인터럽트 서비스 중에 제 1 오퍼레이팅 시스템(OS_A)의 루틴이 수행되었는지 여부를 알리는 정보를 반환하는데, 그러면 그 정보는 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)부터 스레드까지의 실행 시간을 요구한다. 인터럽트 서비스 루틴이 종료된 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 인터럽트 핸들링 작업 때문에 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 수행할 필요가 있으면, 다음에 수행되는 프로그램은 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러 프로그램이며, 그 프로그램 내에서 새로운 상태(READY)가 스레드에 주어진다. 그 스레드에는 인터럽트가 예를

들어 버퍼 내의 정보를 독출하는 것과 같은 조치가 수행되도록 야기한다. 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)는, 우선 순위에 따라서, 실행을 대기하고 있는(READY 상태에서) 제 1 오퍼레이팅 시스템(OS_A)의 스레드들의 실행을 콘트롤 한다.

제 1 오퍼레이팅 시스템(OS_A)의 인터럽트 서비스 상태이거나 또는 실행중인 제 1 오퍼레이팅 시스템(OS_A)의 스레드 내에서, 어떤 상황에서는 지연된 서비스 루틴(DSR)을 형성하는 것이 가능한데, 그 경우 제 2 오퍼레이팅 시스템(OS_B)의 스레드들의 상태가 변화된다. 지연된 서비스 루틴(DSR)은 프로세서(2)가 실행 대기 중이었던 제 1 오퍼레이팅 시스템(OS_A)의 모든 스레드들을 실행한 후에 핸들링된다. 지연된 서비스 루틴(DSR)을 핸들링 한 후에는, 제 2 오퍼레이팅 시스템(OS_B)의 실행을 대기하고 있었던 스레드들을 실행하기 위해 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러 루틴이 우선 순위에 따라서 더 수행되는 것이 바람직하다.

전술된 조치들이 실행되는 동안, 새로운 인터럽트가 발생할 수도 있는데, 그러면 동작은 다시 한번 인터럽트 핸들링 단계로 전송되어 후속 동작들은 전술된 원칙에 따라서 발생된다.

인터럽트 후에 발생하는 동작은, 그 순간에 가장 높은 실질적(effective) 우선 순위를 가지고 있는 동작이 실행되는 방법에 따라서 계속적으로 수행된다. 이러한 동작의 수행은, 예를 들어 프로세서(2)가 인터럽트가 발생하던 순간에 실행 중인 단계에 의하여 영향받고, 또한 인터럽트를 발생시킨 원인(cause)에 의해서도 영향받는다. 첨부된 도 6a 내지 도 6i에 몇 가지 다른 상황들이 예시되어 있는데, 이제부터 이 도면들에 대한 설명이 자세히 진행된다. 내장형 시스템(1)의 일 예는 이동국 펑션 및 데이터 프로세싱 펑션 모두 실장된 통신 장치인데, 이는 개인용 디지털 단말기(PDA)의 펑션들과 같은 것이다. 통신 장치의 프로세서(2) 내에서, 두 개의 오퍼레이팅 시스템들(OS_A, OS_B)이 사용된다. 제 1 오퍼레이팅 시스템(OS_A)은 우선 이동국 펑션들의 콘트롤 및 사용을 위한 것으로 간주된다. 제 2 오퍼레이팅 시스템(OS_B) 는 일반적으로 데이터 프로세싱 펑션의 콘트롤 및 사용을 위한 것으로 간주된다.

도 6a 내지 도 6h에 예시된 상황들에서, 이 상황들은 소위 말하는 정규 인터럽트 리퀘스트(IRQ, normal Interrupt ReQuest)에 관련된다. 본 발명의 바람직한 실시예에 따른 통신 장치(2) 내에서, 소위 말하는 고속 인터럽트 리퀘스트(FIQ, Fast Interrupt Request)가 사용될 수도 있고, 이것은 도 6I에 예시되어 있다. 고속 인터럽트의 우선 순위는 일반적으로 정규 인터럽트(IRQ)의 우선 순위보다 높은 것이 바람직하다. 덧붙이면, 소프트웨어 인터럽트(SWI)역시 사용 가능한데, 이 소프트웨어 인터럽트(SWI)의 우선 순위는 정규 인터럽트(IRQ)의 우선 순위보다 낮은 것이 바람직하다. 첨부된 도 6A 내지 도 6I에 예시된 단계들은 고속 인터럽트(FIQ) 및 프로그램된 인터럽트(SWI)의 경우에 맞게 변형될 수 있으므로 본 명세서에서는 우선적으로 정규 인터럽트(IRQ)의 경우만이 논의된다.

지연된 서비스 루틴(DSR)은 제 2 오퍼레이팅 시스템(OS_B)의 특성(property)인데, 이 특성은 예를 들어 상이한 스레드 간에 메시지 전달 및 필요한 경우, 상이한 오퍼레이팅 시스템들(OS_A, OS_B) 간의 메시지 전달을 위해 사용될 수도 있다. 지연된 서비스 루틴(DSR)은 제 2 오퍼레이팅 시스템(OS_B)의 상태를 실행을 위한 대기 상태(준비 상태, READY 상태)로 변화시킨다. 제 2 오퍼레이팅 시스템(OS_B)의 지연된 서비스 루틴(DSR)의 우선 순위는 스레드들의 우선 순위보다 높기 때문에, 지연된 서비스 루틴(DSR)은 제 2 오퍼레이팅 시스템(OS_B)의 스레드들이 실행되기 이전에 핸들링된다.

프로세서(2)는 이 예시에서 다음의 모드들을 갖는다.

- 사용자 모드(USER)

- 특권 모드(SVC)

- 미결정 모드(UND)

- 고속 인터럽트 모드(FIQ)

- 정규 인터럽트 모드(IRQ), 및

- 소프트웨어 인터럽트 모드(SWI)

비록 본 예시에서 사용된 프로세서(2)에서 디폴트로 사용되는 것은 미결정 모드이지만, 본 발명의 바람직한 실시예에 의한 바람직한 실시예에서는 미결정 모드(UND)는 정규 동작 모드(normal run mode)처럼 사용된다.

도 6a는 정규 인터럽트와 연결되어, 프로세서(2)가 스레드(THB1, 블록 601)를 제 2 오퍼레이팅 시스템(OS_B) 하에 수행하는 상황을 예시하는데, 이는 예를 들어, 통신 장치(1)의 데이터 프로세싱 평션 내에서 사용자에 의해 구동된 응용 프로그램에 관련된 프로세스를 구비한다. 프로세서(2)는 사용자 모드(USER) 내에 있다. 정규 인터럽트가 이 모드를 정규 인터럽트 모드(IRQ)로 전환되게 하는데, 정규 인터럽트 모드(IRQ)에서 프로세서(2)는 이동하여 정규 인터럽트 평션(블록 602)의 첫 부분(begin)을 실행하고, 그 후에는 정규 인터럽트(블록 603)의 서비스 프로그램이 실행된다. 인터럽트에 의하여 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 실행해야할 아무런 이유도 없는데, 그것은 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 인터럽트 후에 호출되지 않기 때문이다. 인터럽트 내에서는 지연된 서비스 리퀘스트(DSR) 역시 형성되지 않는다. 인터럽트 핸들링이 종료되면, 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 그러면 인터럽트가 종료되었다는 정보가 설정된다. 일반 인터럽트의 서비스 프로그램이 종료된 후에는, 프로세서(2)의 모드는 다시 사용자 모드(USER)로 바뀐다. 제 2 오퍼레이팅 시스템(OS_B)의 인터럽트가 발생한 스레드(THB1, 블록 605)가 계속하여 실행된다.

인터럽트는 예를 들어 통신 장치(1)의 뚜껑(lid, 미도시)이 열리는 것에 의하여 발생될 수 있다. 뚜껑이 열리면 뚜껑의 위치를 나타내는 스위치(S1)가 상태를 바꾸는데, 예를 들면, 뚜껑이 열리면, 프로세서(2)는 이 스위치(S1)에 연결된 인식 라인(identification line, 20)의 논리 상태(0/1)를 독출하여 스위치(S1)의 위치를 조사할 수 있다. 이러한 경우에, 뚜껑이 열리는 것 자체는 아무런 동작도 즉시 야기하지는 않는다.

도 6b는 정규 인터럽트와 연결되어, 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A)의 제 1 아이들 스레드(THA_IDLE, 블록 606))를 수행하는 상황을 예시한다. 이러한 상황에서, 모드는 미결정 모드(UND)에서 정규 인터럽트 모드(IRQ)로 변화되고, 프로세서(2)는 정규 인터럽트 평션(블록 602)의 첫 부분을 실행하게 되며, 그 후에는 정규 인터럽트의 서비스 프로그램이 실행된다. 인터럽트에 의하여 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 실행해야할 아무런 이유도 없는데, 그것은 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 인터럽트 후에 호출되지 않기 때문이다. 인터럽트 내에서는 지연된 서비스 리퀘스트(DSR) 역시 형성되지 않는다. 인터럽트가 종료되면, 일반 인터럽트 평션(블록 604)의 종료가 요구되고, 그러면 인터럽트가 종료되었다는 정보가 설정된다. 일반 인터럽트의 서비스 프로그램이 종료된 후에는, 프로세서(2)의 모드는 다시 미결정 모드(UND)로 바뀐다. 아이들 스레드(THA_IDLE, 블록605)로부터 실행이 계속된다.

인터럽트(IRQ)는 예를 들어 통신 장치(1)의 뚜껑(lid)이 열린 위치 상태에서 사용자가 키패드(14)의 키 하나를 누르는 것에 의하여 야기된다. 인터럽트 핸들러가 각 키가 눌린 것이 이 상황에서 이동국 평션에 어떤 효과를 미치는지 여부에 대해서 판단하고, 인터럽트가 이동국 평션에 아무런 조치를 요구하지 않으면 아무 영향도 끼치지 않는다는 것을 확인한다.

도 6c는 일반 인터럽트의 경우에, 프로세서(2)가 스레드(THB1, 블록 601)를 제 2 오퍼레이팅 시스템(OS_B) 하에서 실행하는 상황을 예시하는데, 스레드(THB1)는 예를 들어 사용자에 의하여 구동된 응용 프로그램에 관련된 데이터 프로세싱 평션 내의 프로세스를 구비한다. 이 상황은 또한 화살표 표시(arrow diagram)를 이용하여 도 8에서 역시 예시되어 있다. 프로세서(2)는 사용자 모드(USER)에 있다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 전환되고 프로세스는 정규 인터럽트 평션(블록 602)의 첫 부분을 실행하게 되며, 그 후에는 정규 인터럽트(도 6C의 블록 603, 도 8의 화살표 801 및 802)의 서비스 프로그램이 실행된다. 인터럽트에 의하여 제 1 오퍼레이팅 시스템(OS_A)의 적어도 하나의 스레드를 실행해야 할 필요성이 생기지만, 지연된 서비스 루틴(DSR)은 인터럽트 내에서 형성되지 않는다. 인터럽트가 종료하고 나면(화살표 803), 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 인터럽트의 종료에 대한 정보가 설정된다. 정규 인터럽트의 서비스 프로그램이 종료된 후, 프로세서(2)의 모드는 미결정 모드(UND)로 변화하고 동시에, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 호출된다. 스케줄러(SCH_A)는 필요한 경우 스레드들의 새로운 상태를 설정한다(도 6C의 블록 608, 도 8의 화살표 804). 계속적으로, 실행을 대기하고 있던 제 1 오퍼레이팅 시스템(OS_A)의 스레드들(READY 모드 스레드들)이 실행된다(화살표 805 내지 808). 그 후, 실행되기 위해 대기중인 스레드가 없으면, 프로세서(2)의 모드는 다시 사용자 모드(USER)로 바뀌고 제 2 오퍼레이팅 시스템(OS_B)의 인터럽트가 발생한 스레드(THB1)가 계속하여 실행된다(도 6C의 블록 605, 도 8의 화살표 809, 810 및 811).

이 상황을 예시하는 일 예에서, 응답되지 않은 호출(unanswered call)이 통신 장치(1)에 수신된다. 그러므로, 이동국 평션의 디스플레이 장치(10) 뿐만 아니라 데이터 프로세싱 평션의 디스플레이 장치(15) 모두 수신되고 응답되지 않은 호출에 대한 메시지를 보여준다. 예를 들면 "응답되지 않은 호출 1개"와 같은 메시지를 표시한다. 사용자는 데이터 프로세싱 평션의 엔터 키(미도시)를 누르고, 그러면 정규 인터럽트(IRQ)가 형성된다. 인터럽트 핸들링에서, 전술한 메시지를 이동국 평션의 디스플레이 장치(10)에서 삭제하는 제 1 오퍼레이팅 시스템(OS_A)의 스레드는 실행 모드를 대기하도록 설정된다. 인터럽트가 종료되면, 프로세서(2)는 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)를 실행하게 되는데, 제 1 오퍼레이팅 시스템(OS_A) 의 스레드는 실행 모드에 도달하면 전술한 메시지를 이동국 평션의 디스플레이 장치(10)로부터 삭제한다.

도 6d는, 정규 인터럽트가 발생했을 때 프로세서(2)가 제 2 오퍼레이팅 시스템(OS_B) 하에서 스레드(THB1)를 실행하고 사용자 모드(USER)로 설정되는 상황을 설명한다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 변화하고, 프로세서(2)는 정규 인터럽트 평션(블록 602) 의 시작 부분을 실행하게 되며, 그 후 정규 인터럽트(블록 603)의 서비스 프로그램을 실행한다. 인터럽트에 의하여 제 1 오퍼레이팅 시스템(OS_A)의 스레드 중 적어도 하나를 실행해야 하는 필요성이 생긴다. 덧붙이면, 인터럽트 내에서 및/또는 인터럽트 후에 실행되는 제 1 오퍼레이팅 시스템(OS_A)의 스레드들 내에서 지연된 서비스 루틴(DSR)이 형성된다.

인터럽트가 종료되면, 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 인터럽트의 종료를 나타내는 정보가 설정된다. 정규 인터럽트의 서비스 프로그램이 종료된 후에, 프로세서(2)의 모드는 미결정 모드(UND)로 변화하고, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 호출된다. 스케줄러(SCH_A)는 필요한 경우 스레드에 새로운 상태를 설정하고, 실행되기를 대기하고 있던 제 1 오퍼레이팅 시스템(OS_A)의 스레드들을 스케줄링 프로시져(블록 608)에 맞게 호출한다. 제 1 오퍼레이팅 시스템(OS_A)의 스레드 모두가 실행을 대기하지 않게 되면, 제 2 오퍼레이팅 시스템(OS_B) 의 지연된 서비스 루틴(DSR)이 핸들링되고, 그 후 프로세서(2)의 모드는 다시 사용자 모드(USER)로 변화되고 제 2 오퍼레이팅 시스템(OS_B)의 스레드(THB1, 블록 605)가 다시 실행되기 시작한다. 그 이유는 상태의 지연된 서비스 루틴(delayed service routine of the status)에 의하여 변화된 제 2 오퍼레이팅 시스템(OS_B)의 스레드는 제 2 오퍼레이팅 시스템(OS_B)의 인터럽트가 발생된 스레드(THB1)보다 더 낮은 우선 순위를 갖기 때문이다.

언급될 수 있는 예는, 사용자가 데이터 프로세싱 평션의 키패드(14)를 통해 짧은 메시지(SM, short message)를 입력했을 경우이다. 전송 키(transmission key, 미도시)를 누르면 정규 인터럽트(IRQ)가 발생한다. 인터럽트 핸들링에서, 제 1 오퍼레이팅 시스템(OS_A)의 하나의 스레드 또는 다수 개의 스레드는 전송될 메시지를 버퍼로부터 독출하여 전송하기 위한 실행 모드를 대기하도록 설정되고, 데이터 프로세싱 평션의 디스플레이 장치(15) 상에는 아이콘(icon)이 형성되며, 이 아이콘을 통해 사용자에게 메시지가 전송되었음을 알리게 된다. 인터럽트가 종료되면, 프로세서(2)는 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)를 실행하게 되는데, 그리하여 제 1 오퍼레이팅 시스템(OS_A)의 스레드가 실행 상태에서 메시지를 전송하고, 그 후 지연된 서비스 루틴(DSR)이 형성되어 아이콘을 삭제한다. 계속하여, 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)는 제 2 오퍼레이팅 시스템(OS_B)의 인터럽트가 발생된 스레드(THB1)를 실행하는데, 그 이유는 본 예시에서는 이 스케줄러(THB1)의 우선 순위가 제 2 오퍼레이팅 시스템(OS_B)의 아이콘 삭제 스레드 보다 더 높기 때문이다. 스레드(THB1)가 실행된 후에 아이콘 삭제 스레드는 실행모드가 되고, 데이터 프로세싱 평션의 디스플레이 장치(15)로부터 전술한 텍스트를 삭제한다.

도 6e는 정규 인터럽트가 수신될 때, 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE, 블록 606)를 실행하고 있으며 미결정 모드(UND)로 설정되어 있는 상황에 대한 설명이다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 변화하고, 프로세서(2)는 일반 인터럽트 평션(블록 602)의 시작 부분을 실행하게 되며, 그 후 정규 인터럽트(블록 603)의 서비스 프로그램을 실행한다. 인터럽트에 의하여 제 1 오퍼레이팅 시스템(OS_A)의 스레드 중 적어도 하나를 실행해야하는 필요성이 생긴다. 덧붙이면, 인터럽트 내에서 및/또는 인터럽트 후에 실행되는 제 1 오퍼레이팅 시스템(OS_A)의 스레드들 내에서 지연된 서비스 루틴(DSR)이 형성된다. 이 지연된 서비스 루틴(DSR)이 하나 또는 다수의 제 2 오퍼레이팅 시스템(OS_B)의 스레드의 상태를 변화시켜 실행까지 대기하도록 한다(READY 모드). 인터럽트가 종료되면, 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 인터럽트의 종료를 나타내는 정보가 설정된다. 정규 인터럽트의 서비스 프로그램이 종료된 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 프로세서(2)의 모드는 미결정 모드(UND)로 변화하고, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A, 블록 608)가 요구되는데, 이 스케줄러(SCH_A)가 스케줄러들을 순서대로 실행 모드로 설정한다. 즉 제 1 오퍼레이팅 시스템(OS_A)의 실행을 대기하고 있던 스레드들(READY 모드에서)이 실행된다. 제 1 오퍼레이팅 시스템(OS_A)의 스레드 모두가 실행을 대기하지 않게 되면, 제 2 오퍼레이팅 시스템(OS_B) 의 지연된 서비스 루틴(DSR)이 핸들링되고(블록 609), 그 후 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)가 호출된다(블록610). 스케줄러(SCH_B)는 제 2 오퍼레이팅 시스템(OS_B)의 실행 순서 내에서, 실행을 대기하고 있던 제 1 스레드를 실행 모드로 바꾸는데, 이 경우 순서는 우선 순위에 따르는 것이 바람직하다. 그 후 프로세서(2)의 모드는 다시 사용자 모드(USER)로 변화되고 제 2 오퍼레이팅 시스템(OS_B)의 스레드(THB1)가 다시 실행되기 시작한다(블록 605). 그 이유는 이 스레드(THB1)가 그 순간에 가장 높은 우선 순위를 가지기 때문이다. 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)는 자신의 스케줄링 원칙(scheduling principle)에 따라서 스레드의 실행을 콘트롤한다.

예시로써 제시되는 상황에서, 짧은 메시지가 통신 장치(1) 내에 도착하는데, 이 메시지의 도착은 이동국 평션의 디스플레이 장치(10) 및 데이터 프로세싱 평션의 디스플레이 장치(15) 상의 아이콘에 의해 설명된다. 사용자가 데이터 프로세싱 평션의 키패드(9)를 통해 독출 키(read key, 미도시)을 누르면 정규 인터럽트(IRQ)가 발생한다. 인터럽트 핸들링에서, 제 1 오퍼레이팅 시스템(OS_A)의 하나의 스레드 또는 다수 개의 스레드는 이동국 평션의 디스플레이 장치(10)에서 아이콘을

삭제하기 위하여 및 이동국 평션의 디스플레이 장치(10) 상에 수신된 메시지를 표시하기 위하여 실행 모드까지 대기하도록 설정된다. 또한, 하여, 지연된 서비스 루틴(DSR)이 형성되어 데이터 프로세싱 장치의 디스플레이 장치(15)로부터 아이콘을 삭제한다. 프로세서(2)는 인터럽트 후에 제 1 오퍼레이팅 시스템(OS_A)의 스케줄링 평션을 실행하고, 지연된 서비스 루틴(DSR)을 핸들링하고, 그리고 제 2 오퍼레이팅 시스템(OS_B)의 스케줄링 평션을 실행하는데, 그러면 제 2 오퍼레이팅 시스템(OS_B)의 스레드가 실행 모드가 된 후에, 데이터 프로세싱 평션의 디스플레이 장치(15) 상에서 아이콘을 삭제한다.

도 6f는 정규 인터럽트가 발생했을 때 프로세서(2)가 제 2 오퍼레이팅 시스템(OS_B) 하에서 스레드(THB1)를 실행하고 사용자 모드(USER)로 설정되는 상황을 설명한다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 변화하고, 프로세서(2)는 일반 인터럽트 평션(블록 602) 의 시작 부분을 실행하고, 그 후 정규 인터럽트(블록 603)의 서비스 프로그램을 실행한다. 인터럽트는 제 1 오퍼레이팅 시스템(OS_A)의 스레드 중 적어도 하나를 실행해야하는 필요성을 야기한다. 뿐만 아니라, 인터럽트 내에서 및/또는 인터럽트 후에 실행되는 제 1 오퍼레이팅 시스템(OS_A)의 스레드들 내에서 지연된 서비스 루틴(DSR)이 형성되고, 이 지연된 서비스 루틴(DSR)이 제 2 오퍼레이팅 시스템(OS_B)의 스레드들의 실행 시간을 리셋 하도록 야기한다. 인터럽트가 종료되면, 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 인터럽트의 종료를 나타내는 정보가 설정된다. 정규 인터럽트의 서비스 프로그램이 종료된 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 프로세서(2)의 모드는 미결정 모드(UND)로 변화하고, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 호출되고 호출된 스케줄러(SCH_A)는 필요한 경우 스레드에 새로운 상태를 설정한다(블록 608). 그 이후에, 실행되기를 대기하고 있던 제 1 오퍼레이팅 시스템(OS_A)의 스레드들(READY 상태에 있는 스레드들)이 실행된다. 제 1 오퍼레이팅 시스템(OS_A)의 스레드 중 실행을 대기하는 스레드가 존재하지 않게 되면, 제 2 오퍼레이팅 시스템(OS_B) 의 지연된 서비스 루틴(DSR)이 핸들링되고(블록 609), 그러면 제 2 오퍼레이팅 시스템(OS_B)의 스케줄러(SCH_B)가 호출된다(블록 610). 호출된 스케줄러(SCH_B)는 제 2 오퍼레이팅 시스템(OS_B)의 스레드로써 실행 대기 순서 중 맨 앞에 위치하고 있던 스레드를 실행 모드로 실행시킨다. 본 예시에서, 인터럽트 핸들링에 의하여, 인터럽트가 발생한 스레드(THB1)는 아직은 실행 모드로 이동해 갈 순서가 아니고, 그 대신에 상태가 변화된 제 2 오퍼레이팅 시스템(OS_B)의 다른 스레드(THB2)가 실행을 대기하도록 야기된다. 프로세서(2)의 모드는 다시 사용자 모드(USER)로 변화되고, 실행 모드(블록 611)로 설정된 제 2 오퍼레이팅 시스템(OS_B)의 스레드(THB2)가 실행되기 시작한다. 인터럽트가 발생된 스레드(THB1)는 자기 순서일 때 실행 모드에 도달한다(블록 605).

전술된 예시는 사용자가 데이터 프로세싱 평션의 키패드(14)를 이용하여 짧은 메시지를 기입하고 충전 장치(charging device, 미도시)를 통신 장치(1)에 연결하는 상황이다. 그러면 정규 인터럽트(IRQ)가 발생한다. 인터럽트 핸들링에서, 충전이 수행된다는 것을 알리는 아이콘을 이동국 평션의 디스플레이 장치(10) 상에 형성하는 제 1 오퍼레이팅 시스템(OS_A) 의 스레드가 실행 모드를 대기하도록 설정된다. 더 나아가, 지연된 서비스 루틴(DSR)을 이용하여, 이에 상응하는 아이콘이 데이터 프로세싱 평션의 디스플레이 장치(15) 상에도 형성된다.

도 6g는 정규 인터럽트가 발생했을 때, 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A) 하에서 스레드(THA1)를 실행하고 있는 상황(블록 612)을 예시한다. 스레드(THA1)는 예를 들면 통신 장치(1)의 연결-설정(connection-establishing) 응용 프로그램이다. 프로세서(2)는 미결정 모드(UND)에 있다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 변화되고 프로세서(2)는 이동하여 정규 인터럽트 평션의 시작 부분을 실행하고(블록 602) 그 뒤에 정규 인터럽트의 서비스 프로그램(블록 603)이 실행된다. 인터럽트는 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 실행해야할 어떤 필요성도 야기하지 않는다. 인터럽트 후에, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 호출되지 않는다. 인터럽트가 종료된 후, 정규 인터럽트 평션의 종료가 호출되고(블록 604), 인터럽트의 종료에 대한 정보가 설정된다. 정규 인터럽트의 종료 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 프로세서(2)의 모드는 다시 미결정 모드(UND)로 변화되고 인터럽트가 발생된 스레드(THA1)부터 실행이 계속된다(블록 613). 비록 제 2 오퍼레이팅 시스템(OS_B) 의 지연된 서비스 루틴(DSR)이 인터럽트 내에서 형성되기는 하지만, 이 것은 제 1 오퍼레이팅 시스템(OS_A)의 모든 실행 중인 스레드가 수행되기 전에는 핸들링되지 않는다.

사용자는 예를 들어 전화 번호를 다이얼 하는데, 키를 누르면 정규 인터럽트(IRQ)를 야기한다. 인터럽트 핸들링에서, 키를 누르는 것도 독출되어 메모리(17) 내로 기입되고 이동국 평션의 디스플레이 장치(10)로부터 출력되며, 귀환되어 다음번 키가 눌려질 때까지 대기된다.

도 6h는 정규 인터럽트가 발생했을 때, 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A) 하에서 스레드(THA1)를 실행하고 있는 상황(블록 612)을 예시한다. 스레드(THA1)는 예를 들면 통신 장치(1)의 연결-설정(connection-establishing) 응용 프로그램이다. 프로세서(2)는 미결정 모드(UND)에 있다. 그러므로, 모드는 정규 인터럽트 모드(IRQ)로 변화되고 프로세서(2)는 이동하여 정규 인터럽트 평션의 시작 부분을 실행하고(블록 602) 그 뒤에 정규 인터럽트의 서비스 프로그램(블록 603)이 실행된다. 인터럽트는 제 1 오퍼레이팅 시스템(OS_A)의 스레드를 적어도 하나 실행해야할 필요성을 야기한다. 인터럽트가 종료된 후에, 정규 인터럽트 평션(블록 604)의 종료가 호출되고, 인터럽트의 종료에 대한 정보가 설정된다. 정규

인터럽트의 서비스 프로그램이 종료된 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 프로세서(2)의 모드는 다시 미결정 모드(UND)로 변화되고 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 요구된다. 스케줄러(SCH_A)는 필요할 경우 스레드를 위해 새로운 상태를 설정한다(블록 608). 결과적으로, 실행을 대기하는(READY 모드에서) 제 1 오퍼레이팅 시스템(OS_A)의 스레드들은 우선 순위에 따라서 실행된다. 예시에서, 인터럽트는 제 2 스레드(THA2)가 즉시 이동하여 실행되도록 한다(블록 614). 그러면 인터럽트가 발생한 스레드(THA1)가 아마도 차후에 실행된다(블록 613). 비록 제 2 오퍼레이팅 시스템(OS_B) 의 지연된 서비스 루틴(DSR)이 인터럽트 내에서 형성되기는 하지만, 이 것은 제 1 오퍼레이팅 시스템(OS_A)의 모든 실행 중인 스레드가 수행되기 전에는 핸들링되지 않는다.

사용자는 예를 들어 전화 번호를 다이얼하고 핸드셋 업 키(handset-up key, 미도시)를 누르는데, 그러면 정규 인터럽트(IRQ)가 발생된다. 인터럽트 핸들링에서, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 연결-설정 스레드를 실행 상태로 설정한다.

도 6i는 정규 인터럽트 서비스 루틴이 실행되는 중에(블록 615) 고속 인터럽트가 발생하는 상황을 예시한다. 프로세서(2)는 정규 인터럽트 모드(IRQ) 내에 있고 이동하여 고속 인터럽트 모드(FIQ)로 이동한다. 프로세서(2)는 정규 인터럽트 펑션의 시작 부분을 실행하고(블록 616) 그 뒤에 고속 인터럽트(FIQ)의 서비스 프로그램(블록 617)이 실행된다. 고속 인터럽트가 종료된 후에, 고속 인터럽트 펑션(블록 604)의 종료가 호출되고, 고속 인터럽트(FIQ)의 종료에 대한 정보가 설정된다. 고속 인터럽트의 서비스 프로그램이 종료된 후에, 프로세서(2)의 모드 및 제 1 오퍼레이팅 시스템(OS_A)의 상태가 변화된다. 프로세서(2)의 모드는 다시 정규 인터럽트 상태(IRQ)로 변화되고 인터럽트가 발생했던 위치부터(블록 619) 동작이 계속된다.

이러한 상황에 대한 예를 제공하기 위하여, 프로세서(2)는 예를 들어 응용 프로그램에 특유한 집적 회로(3)와 같은 통신 장치(1) 내에 설치된 타이머(미도시)가 고속 인터럽트(FIQ)를 형성할 때 키패드(9, 14)의 인터럽트 핸들링을 실행하는 상황의 예시이다. 프로세서(2)는 타이머의 값을 독출하고 이것을 메모리(17) 내에 저장한다. 저장 후, 프로세서(2)는 키패드(9, 14)의 인터럽트 핸들링으로 돌아간다.

프로세서(2)의 전술된 모드 변화와 관련하여, 프로세서(2)는 상이한 레지스터들을 사용하여 시동되는 것이 바람직한데, 이 경우 데이터를 임시 저장 장소에 전송(transfer)하는 것이 언제나 필요한 것은 아니다. 그러나, 어떤 인터럽트 상태에서는, 데이터 전송이 필요한데, 이 경우 데이터 전송이 인터럽트를 핸들링하는 것을 어느 정도 배제(retard)한다. 또한, 프로세서(2)로 사용되는 어떤 마이크로 프로세서들은 전술된 모든 모드들을 구비하는 것은 아니고, 이 경우 전술된 상이한 모드들은 모드 변수(mode variable) 또는 그와 같은 것들을 이용하여 나타낼 수 있다.

가끔 제 2 오퍼레이팅 시스템(OS_B)에게도 응답 시간을 결정하는 것이 필요할 수 있다. 그러므로, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)는 제 2 오퍼레이팅 시스템(OS_B)을 위해 남아 있는 실행 시간(execution time)이 응답 시간(response time)이 요구하는 시간보다 적은 상황에서 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)의 우선 순위를 높일 수 있다. 그러므로, 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)는 다른 중요하지 않은(non-critical) 스레드들 보다 빨리 실행 순서가 되는데, 그러므로 제 2 오퍼레이팅 시스템(OS_B)이 실행되도록 전송을 야기한다. 이처럼 우선 순위 레벨을 높이는 것은 예를 들면 소정 시간 후에 인터럽트를 발생하도록 설정된 타이머 등에 의하여 구동된다. 타이머 인터럽트가 도달했을 때 프로세서(2)가 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)에 도달하지 않으면, 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 호출된다. 그러면 아이들 스레드(THA_IDLE)의 우선 순위가 증가된다. 하지만, 이 경우에 아이들 스레드(THA_IDLE)의 우선 순위를 짧은 시간 동안에만 높여서 제 1 오퍼레이팅 시스템(OS_A)의 중요하지 않은(non-critical) 작업들에도 일반적으로 실행 시간(execution time)이 제공되도록 하는 것이 바람직하다. 제 1 오퍼레이팅 시스템(OS_A)의 중요한 스레드(critical thread)가 실행되어야 하기 때문에, 아이들 스레드(THA_IDLE)의 우선 순위를 너무 높게 올릴 수는 없다. 이러한 종류의 스케줄링은 삽입형 스케줄링(interleaved scheduling)이라고 불릴 수 있다.

첨부된 도 9 및 도 10은 전술된 삽입형 스케줄링 옵션의 일 예를 설명한다. 도 9에서는 제 1 오퍼레이팅 시스템(OS_A) 내에서 동작하는 프로세스들의 실행 스레드들(THA1, THA2, THA3) 및 제 2 오퍼레이팅 시스템(OS_B)의 실행 스레드들(THB1, THB2, THB3)이 실행되어야 하는 예시 상황의 타이밍도가 설명된다. 도 10에서는 삽입형 스케줄링을 더 명확히 하기 위한 제 1 오퍼레이팅 시스템(OS_A)의 프로세스 도(process diagram)가 설명된다.

예시로 제시된 도면에서 제 1 오퍼레이팅 시스템(OS_A)에서 실행되는 프로세스들의 세 개의 스레드들(THA1, THA2, THA3) 및 제 2 오퍼레이팅 시스템(OS_B)에서 실행되는 프로세스들의 세 개의 스레드들(THB1, THB2, THB3)이 있지만, 오퍼레이팅 시스템들(OS_A, OS_B) 내에는 더 많은 수의 또는 더 적은 수의 스레드들이 존재할 수 있다는 것이 이해될 것이다.

제 1 오퍼레이팅 시스템(OS_A)의 제 1 스레드(THA1)는 예를 들면 전화기 응용 프로그램이고 매우 높은 우선 순위를 갖는다. 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)는 예를 들면 사용자 인터페이스(user interface)를 위하여 제공되었다. 제 1 오퍼레이팅 시스템(OS_A)에서, 제 2 스레드(THA2)의 우선 순위는 제 1 스레드(THA1)의 우선 순위보다는 낮고, 제 3 스레드(THA3)의 우선 순위는 제 2 스레드(THA2)의 우선 순위보다는 낮다.

제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)는 예를 들면 인터페이스 블록(interface block, 16)을 통한 통신을 위한 프로그램이다. 제 2 오퍼레이팅 시스템(OS_B)의 제 2 스레드(THB2) 및 제 3 스레드(THB3)를 예를 들면 어떤 개인용 디지털 단말기(PDA) 응용 프로그램으로써 예를 들면 메모장 응용 프로그램 및 달력 응용 프로그램이다. 이러한 예시에서, 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)의 우선 순위는 제 2 오퍼레이팅 시스템(OS_B)의 제 2 스레드(THB2)의 우선 순위보다는 높고 제 2 오퍼레이팅 시스템(OS_B)의 제 3 스레드(THB3)의 우선 순위는 제 1 내지 제 3 스레드들 중에서 가장 낮다는 것이 전제된다.

도 9에서, 스레드들이 실행되는 것은 굵은 선으로 나타냈다. 예를 들어, 시점(t1)에서 제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)는 실행 순서가 되었다. 이것은 참조 번호 901번으로 표시된다. 시점(t1) 보다 늦은 시점(t2)에서 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)는 실행 순서가 된다(902번). 제 3 스레드(THA3)가 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE) 이전에 마지막 스레드라는 것이 역시 가정된다. 그러므로, 잠시 후에 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)가 시점(t3)에서 실행 순서가 되어(903) 인터페이스 블록(15)을 통해서 통신할 필요가 있는지를 조사한다. 더 늦은 시점(t4)에서, 제 2 오퍼레이팅 시스템(OS_B)의 제 2 스레드(THB2)가 실행 순서가 된다(904). 또한 제 2 오퍼레이팅 시스템(OS_B)의 제 3 스레드(THB3)가 실행되기 위한 시점(t5)이 존재한다. 이 예시에서 다음에 실행 순서(906)가 될 스레드는 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)로서 시점(t6)에 실행된다.

전술한 상황이 도 10에 예시되는데, 시점(t7) 이후부터 도시되어 있다. 스레드들(THA1, THA2, THA3), 스케줄러(SCH_A), 인터럽트 서비스 루틴(ISR), 인터럽트 벡터(interrupt vectors) 및 제 1 오퍼레이팅 시스템(OS_A)의 다른 서비스들은 도 10에 블록으로 제시되었다. 도 10 내의 스레드들의 우선 순위는 가장 높은 스레드가 가장 높은 우선 순위를 갖도록 구현된다. 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)가 가장 낮은 우선 순위를 갖는다. 시점(t7)에 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)가 제 1 오퍼레이팅 시스템의 아이들 스레드(THA_IDLE)의 우선 순위를 잠시동안 변화시킨다. 이러한 상황이 도 10 내에서 시점(t7) 바로 직후부터 설명되어 있다. 시점(t11)에서 제 1 오퍼레이팅 시스템(OS_A)의 스케줄러(SCH_A)는 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)의 우선 순위를 다시 가장 낮은 순위로 바꾼다. 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)의 우선 순위가 변화되는 것은 예를 들어 타이며, 다른 사건 들을 이용하여 구현될 수 있다.

예를 들어 사용자는 제 2 인터페이스(UI2)의 몇 개의 키를 누른다. 그러면, 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)가 다시 시점(t8)에서 실행 순서(907)가 된다. 스케줄러(SCH_A) 역시 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)의 우선 순위를 높인다. 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)는 인터페이스 블록(16)을 통해 전송할 필요가 있는 동안 실행된다. 이것은 인터럽트를 발생시키고, 시점(t9)에서 아이들 스레드(THA_IDLE)의 우선 순위가 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)보다 높기 때문에, 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)는 인터럽트 되고 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)는 실행 순서(908)가 된다. 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)를 실행한 후, 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THB3)가 시점(t10)에서 계속하여 실행된다(909).

스케줄러(SCH_A)가 제 1 오퍼레이팅 시스템(OS_A)의 아이들 스레드(THA_IDLE)의 우선 순위를 다시 낮춰, 이 예시의 시점(t11)에서 가장 낮은 우선 순위 레벨로 만든다.

제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)를 실행하는 것은 시점(t12)에서 인터럽트 되는데, 그것은 실행(910)을 알리는(signalled) 보다 높은 우선 순위를 갖는 스레드가 있기 때문이다. 예를 들어서, 이 스레드는 제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)이다.

시점(t13)에서, 제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)를 실행하는 것은 인터럽트 되는데, 그 이유는 더 높은 우선 순위를 갖는 스레드가 있기 때문이다. 제 1 오퍼레이팅 시스템(OS_A)의 제 1 스레드(THA1)는 실행(911)을 위하여 표시(signalled)된다.

제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)를 실행하는 것은(912) 시점(t14)에서 제 1 오퍼레이팅 시스템(OS_A)의 제 1 스레드(THA1)가 실행된 후에 계속된다.

시점(t15)에서 제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)는 완료된다. 동일한 시점(t15)에 또한 제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)는 실행될 준비가 된다. 그러나, 제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)를 실행하는 것은 시점(t15)에서부터 계속된다(913). 이것은 시점(t9)에서의 전 동작과는 반대인 것이다.

제 1 오퍼레이팅 시스템(OS_A)의 제 3 스레드(THA3)를 실행하는 것(913)은 제 1 오퍼레이팅 시스템(OS_A)의 제 2 스레드(THA2)가 시점(t15)에서 실행된 후에 계속된다.

제 2 오퍼레이팅 시스템(OS_B)의 제 1 스레드(THB1)는 시점(t16)에서 실행 순서(914)가 되고 인터페이스 블록(16)을 통해 서비스 전송을 가능하게 한다.

표 1은 인터럽트와 연결되어 하나의 모드에서 다른 모드까지 전송되는 현상을 간략화 한 것이다. 제 1 수직 라인은 모드가 변화할 때 프로세서(2)의 모드를 예시한다. 제 1 수평 라인은 프로세서(2)가 전송되는 모드이다. 표의 다른 박스들은 각 모드들 간의 변화를 야기하는 상황들에 대한 예시이다. 약자로 사용된 것은 본 명세서의 앞부분에서 상이한 모드들 및 상이한 인터럽트 간에 사용되었던 약자들을 의미한다.

[표 1]

	USER	UND	SVC	IRQ	FIQ
USER	...	...	OS_B 스레드가 SWI를 요구	IRQ 가 OS_B 스레드를 인터럽트	FIQ가 OS_B 스레드를 인터럽트
UND	OS_A 스레드 중 최종 스레드(last thread) 또는 DSR 로부터 귀환	....	DSR 또는 아이들 스레드가 SWI를 요구	IRQ 가 OS_A 스레드, DSR 또는 아이들 스레드(idle thread)를 인터럽트	FIQ가 OS_A 스레드, DSR 또는 아이들 스레드를 인터럽트
SVC	고속 SWI로부터 귀환	저속 SWI 동안	....	IRQ 가 고속 SWI를 인터럽트	FIQ가 고속 SWI를 인터럽트
IRQ	활성화된 OS_A 스레드 또는 DSR이 존재하지 않으면, IRQ로부터 귀환	활성화된 OS_A 스레드 또는 DSR이 존재하면, IRQ로부터 귀환	....	....	FIQ가 IRQ를 인터럽트
FIQ	활성화된 OS_A 스레드 또는 DSR이 존재하지 않으면, FIQ로부터 귀환	활성화된 OS_A 스레드 또는 DSR이 존재하면, FIQ로부터 귀환	....	IRQ가 인터럽트 되면, FIQ로부터 귀환	.....

다음의 예시는 상이한 오퍼레이팅 시스템(OS_A, OS_B)에서 실행되는 스레드들 간에 메시지를 전송하는 것을 설명한다. 도 7a 및 도 7b는 간략화된 다이어그램으로 메시지 전송을 예시한다. 메시지 드라이버(MD) 뿐만 아니라 메시지 대기 행렬(message queue, MQ1, MQ2)은 오퍼레이팅 시스템(OS_A, OS_B) 모두에 공통된다. 제 2 오퍼레이팅 시스템(OS_B)의 실행되는 스레드(running thread, THB1)는 메시지 드라이브(MD)를 호출하고 예를 들어 텍스트 메시지와 같은 전송 메시지 및 목적 스레드(object thread, THA1)를 파라미터로서 전송하는 것이 바람직하다(도 7a의 화살표 701). 메시지 드라이버(MD)는 제 1 메시지 대기 행렬(MQ1)로 메시지를 전송하고(화살표 702) 소프트웨어 인터럽트(SWI)를 형성하는 것이 바람직한데, 이 인터럽트는 실질적으로 본 명세서에서 전술된 도 6c에 따라서 인터럽트 핸들링을 구동한다. 다만 이 인터럽트가 정규 인터럽트(IRQ) 대신에 소프트웨어 인터럽트(SWI)에 의해 야기된 점에 전술된 바와 다르다. 인터럽트 핸들러를 떠나면 제 1 오퍼레이팅 시스템(OS_A)의 스레드(SCH_A)가 구동되는데, 목적 스레드는 우선 순위에 의하여 결정된 단계로 실행된다(화살표 703). 이 목적 스레드(THA1)가 제 1 메시지 대기 행렬(MQ1) 내에서 메시지를 독출하고 예를 들어 무선 경로(radio path)로 메시지의 전송을 전송하는 것을 실행한다.

제 1 오퍼레이팅 시스템(OS_A)의 응용 프로그램에서, 지연된 서비스 루틴(DSR)이 실행될 수 있는데, 이 지연된 서비스 루틴(DSR)을 통해서 예를 들어 제 2 오퍼레이팅 시스템(OS_B)의 스레드들로 메시지를 전송하는 것이 가능하다. 그 전송 방법을 예를 들면 다음과 같다(도 7b). 제 2 오퍼레이팅 시스템(OS_B)에서, 실행중인 스레드(THB2)는 메시지 리퀘스트(message request)를 형성하고 메시지를 대기하도록 남는다(화살표 705). 메시지 리퀘스트는 소프트웨어 인터럽트(SWI)를 야기하고, 인터럽트 핸들러가 인터럽트의 원인을 체크하고 제 2 오퍼레이팅 시스템(OS_B)의 스레드(THB2)가 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA2)로부터 메시지를 전송 받기를 대기하고 있다고 결정한다. 인터럽트 핸들링 이후에, 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA2)가 호출된다(화살표 706). 원하는 스레드(THA2)가 실행되면, 이것은 메시지를 형성하고 형성된 메시지를 제 2 메시지 대기 행렬(MQ2)에 추가한다(화살표 707). 더 나아가, 스케줄러(THA2)는 지연된 서비스 루틴(DSR)을 형성한다(화살표 708). 제 1 오퍼레이팅 시스템의 스레드가 실행된 후에, 이 지연된 서비스 루틴(DSR)이 핸들링되는데, 스케줄러(THA2)에 의하여 추가된 지연된 서비스 루틴(DSR)이 제 2 오퍼레이팅 시스템(OS_B)의 스레드의 상태를 실행 준비 상태(READY 상태)로 변화시킨다(화살표 709). 제 2 오퍼레이팅 시스템(OS_B)의 스레드(SCH_B)는 자신의 스케줄링 원칙(scheduling principle)에 따라서 스레드들을 실행한다. 전송 리퀘스트(transmission request)를 발생한 스레드(THB2)가 실행되는 동안(화살표 710), 이것은 제 2 메시지 대기 행렬(MQ2)로부터 메시지를 독출하여 메시지 드라이버(MD)의 접속 인터페이스를 통하여 전송한다.(화살표 711).

실제 실시예에서, 메시지 대기 행렬(MQ1, MQ2)은 우선 순위에 따라 구성되는 것이 바람직하다. 이것은 높은 우선 순위를 갖는 수신 스레드(receiving thread)의 메시지가 낮은 우선 순위를 가지고 대기 행렬 메시지(queue message) 내를 지난다는 것을 의미한다. 또는, 본 예시에서 사용된 제 1 메시지 대기 행렬(MQ1) 및 제 2 메시지 대기 행렬(MQ2)보다 더 많은 메시지 대기 행렬들이 사용될 수 있다.

산업상 이용 가능성

요약하면, 본 발명은 두 개 또는 그 이상의 오퍼레이팅 시스템들(OS_A, OS_B)이 단일 프로세서 내에 사용하는 것을 가능하게 한다는 것이 주목될 수 있다. 종래 기술에 의한 응용 프로그램이 통신 장치(1) 내에서 사용될 수 있는 데, 이 경우 본 발명의 하나의 목적은 공통 부분(common part)을 최소화 하는 것이다. 오퍼레이팅 시스템들(OS_A, OS_B)의 공통 부분은 일반적으로 인터럽트 핸들러들인데, 인터럽트 핸들러 내에서 오퍼레이팅 시스템들(OS_A, OS_B)로의 접속(connection)이 형성된다. 또한 메시지 드라이버(ME)가 전술한 예시에서 공통 부분이다.

실용적인 내장형 시스템(1) 내에서 상이한 오퍼레이팅 시스템들(OS_A, OS_B)을 위하여 다수의 공통 인터럽트 핸들러들(인터럽트 서비스 루틴들)이 배치될 수 있다는 것이 가능하다. 하나의 인터럽트 당 다수의 인터럽트 핸들러 중 하나만이 한번에 활성화될 수 있다, 즉 예를 들어 고속 인터럽트(FIQ)를 위한 하나의 인터럽트 핸들러, 정규 인터럽트(IRQ)를 위해 하나의 인터럽트 핸들러, 또 소프트웨어 인터럽트(SWI)를 위해 하나의 인터럽트 핸들러가 활성화 될 수 있다. 그러므로, 상기 인터럽트를 위해 배치된 다른 하나의 인터럽트 핸들러를 변화시켜 활성화된 인터럽트 핸들러로 바꾸는 것이 가능하다.

단일 프로세서(2) 시스템 내의 두 개의 오퍼레이팅 시스템들(OS_A, OS_B)의 동작이, 예를 들어 제 2 오퍼레이팅 시스템(OS_B) 의 스레드들(THB1, THB2, THB_IDLE) 중에서 제 1 오퍼레이팅 시스템(OS_A)의 스레드가 형성되는 방법으로 실장될 수 있다는 것이 언급되어야만 한다. 형성되는 제 1 오퍼레이팅 시스템(OS_A)의 스레드는 아이들 스레드(THA_IDLE) 또는 수퍼 스레드 같은 것인데, 제 2 오퍼레이팅 시스템(OS_B) 와 관련된 모든 실행 스레드들(THB1, THB2, THB_IDLE)이 이 스레드 내에서 실행된다. 또한, 제 1 오퍼레이팅 시스템(OS_A) 의 스레드들(THA1, THA2, THA_IDLE) 중에서 제 2 오퍼레이팅 시스템(OS_B)의 스레드가 형성되는 방법으로 실장되는 것도 가능하다. 형성되는 제 2 오퍼레이팅 시스템(OS_B)의 스레드는 아이들 스레드(THA_IDLE) 또는 수퍼 스레드(511) 같은 것인데, 제 1 오퍼레이팅 시스템(OS_A) 와 관련된 모든 실행 스레드들(THA1, THA2, THA_IDLE)이 이 스레드 내에서 실행된다. 또다른 대안은 제 1 오퍼레이팅 시스템(OS_A)의 각 스레드(THA1, THA2, THA_IDLE)가 제 2 오퍼레이팅 시스템(OS_B)에 개별적인 수퍼 스레드로서 형성되는 것인데, 제 2 오퍼레이팅 시스템(OS_B)은 이 스레드들을 분리된 수퍼 스레드(미도시) 처럼 핸들링한다. 본 명세서에서 전술된 바와 같이, 이러한 수퍼 스레드들의 응답 시간은 전형적으로는 정규 스레드들의 응답 시간에 비하여 훨씬 작다. 가장 중요한 응답 시간 요구(most critical response time requirement)를 가지는 작업은 여전히 그들과 예를 들어 타이머의 인터럽트 핸들러와 접속하여 인터럽트 핸들링을 형성함으로써 실행될 수 있다. 이 때, 스케줄링 펑션은 이 작업들을 수행하기 위하여 이동할 때 반드시 사용되는 것은 아니다.

제 1 오퍼레이팅 시스템(OS_A)의 각 스레드(THA1, THA2, THA_IDLE)가 제 2 오퍼레이팅 시스템(OS_B)의 개별 수퍼 스레드(511)로서 실장된 경우에는, 제 1 오퍼레이팅 시스템(OS_A)의 스레드(THA1, THA2, THA_IDLE) 간에 데이터를

전송할 필요가 있을 때 제 2 오퍼레이팅 시스템(OS_B)에 따른 신호 전송이 요구된다. 반면에, 제 1 오퍼레이팅 시스템(OS_A)의 스레드들(THA1, THA2, THA_IDLE)이 단일 수퍼 스레드(511) 처럼 실장되면, 데이터 전송은 제 1 오퍼레이팅 시스템(OS_A)의 메시지 전송 방법을 이용하여 훨씬 용이한 방법으로 실장될 수 있다.

본 발명은 전술된 실시예들에 한정되는 것이 아니며, 첨부된 청구 범위의 기술적 사상의 범위 내에서 변형될 수 있다.

도면의 간단한 설명

이하, 본 발명은 첨부된 도면을 참조하여 더 자세히 설명될 것이다. 본 명세서에 첨부된 도면들은 다음과 같다.

도 1은 단일 오퍼레이팅 시스템의 계층 구조(layer structure)를 예시하는 도면이다.

도 2는 프로세스(process)들의 상태 모델(state model)의 하나를 예시하는 도면이다.

도 3은 프로세스 요소(process element)의 한 예를 예시하는 도면이다.

도 4a는 본 발명의 바람직한 실시예에 의한 내장형 시스템의 간략화된 블록도이다.

도 4b는 단일 프로세서의 간략화된 블록도이다.

도 5는 본 발명의 내장형 시스템과 연관되어 사용되는 한 오퍼레이팅 시스템 아키텍쳐(architecture)의 간략화된 도면이다.

도 6a 내지 도 6i는 본 발명의 바람직한 실시예에 의한 내장형 시스템 내에서 인터럽트를 핸들링하는 상이한 상황(situation)을 예시하는 도면들이다.

도 7a 및 도 7b는 메시지 전송 메커니즘의 간략화된 도면이다.

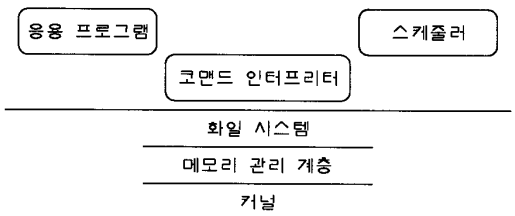
도 8은 본 발명의 바람직한 실시예에 의한 내장형 시스템 내에서 인터럽트를 핸들링하는 상이한 상황을 예시하는 간략화된 도면들이다.

도 9는 본 발명의 바람직한 실시예에 의한 내장형 시스템 내의 삽입형 스케줄링 옵션(interleaved scheduling option)의 타이밍도를 나타내는 도면이다.

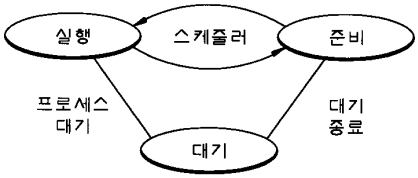
도 10은 본 발명의 바람직한 실시예에 의한 내장형 시스템 내의 삽입형 스케줄링 옵션의 동작을 나타내는 도면이다.

도면

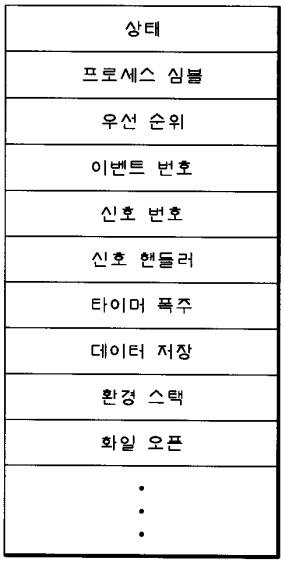
도면1



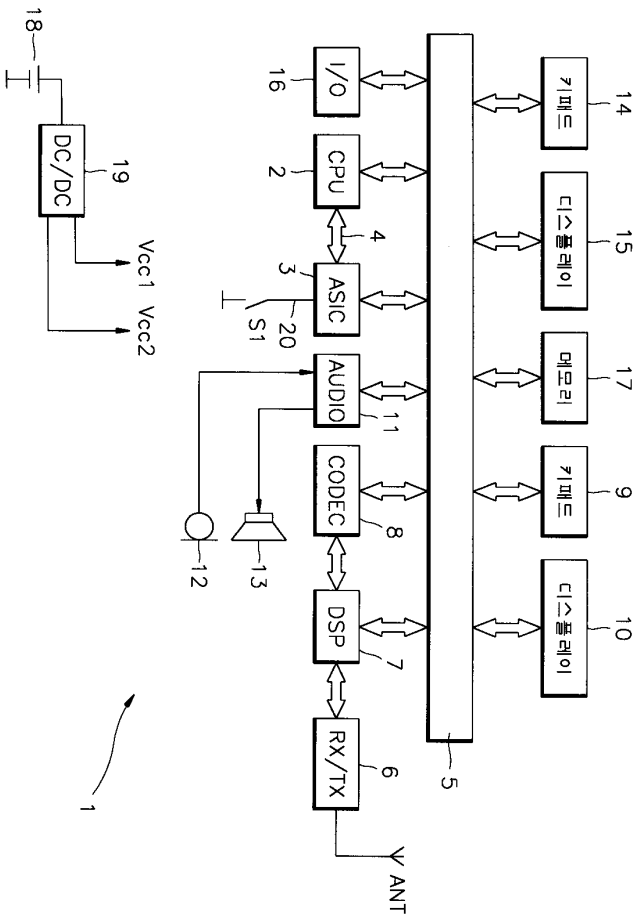
도면2


실행
스케줄러
준비
프로세스
대기
대기
종료
대기

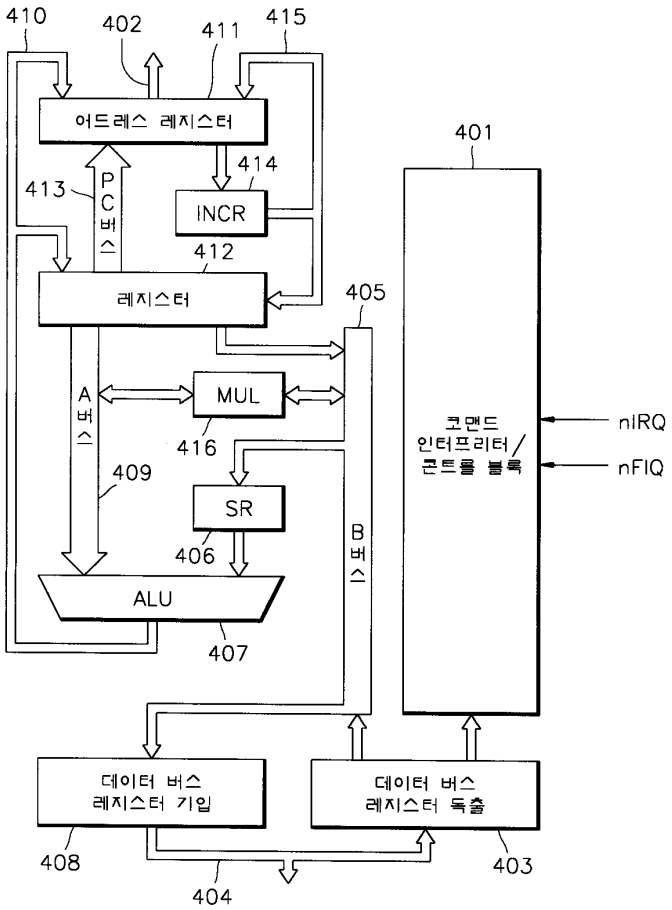
도면3


상태
프로세스 심볼
우선 순위
이벤트 번호
신호 번호
신호 핸들러
타이머 폭주
데이터 저장
환경 스택
화일 오픈

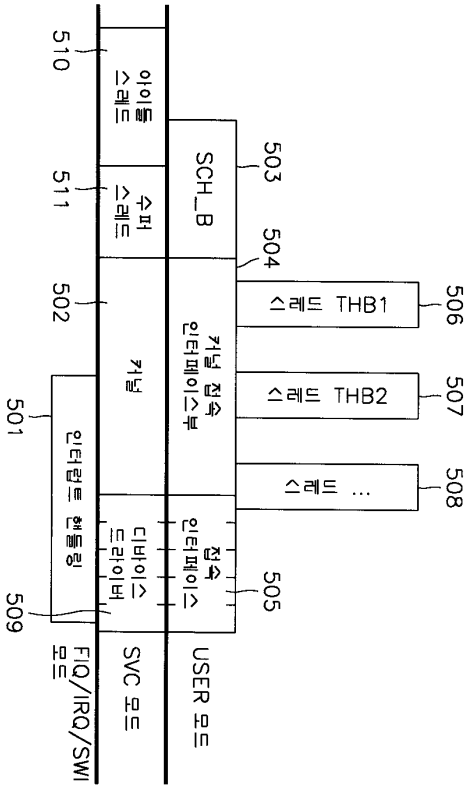
도면4a


18
DC/DC
19
Vcc1
Vcc2
1
I/O
16
CPU
2
4
ASIC
3
S1
20
AUDIO
11
CODEC
8
DSP
7
RX/TX
6
ANT
12
13
5
키패드
14
디스플레이
15
메모리
17
키패드
9
디스플레이
10

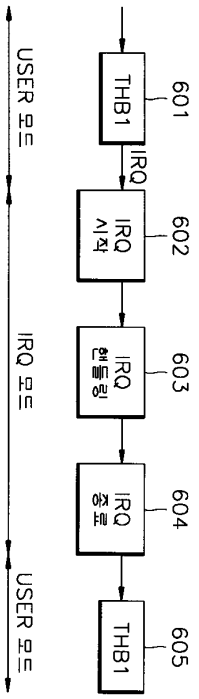
도면4b


410
402
411
415
어드레스 레지스터
401
414
413
PC 버스
INCR
412
레지스터
405
A 버스
MUL
416
409
SR
406
B 버스
ALU
407
코맨드 인터프리터/ 콘트롤 블록
nIRQ
nFIQ
데이터 버스 레지스터 기입
데이터 버스 레지스터 독출
408
404
403

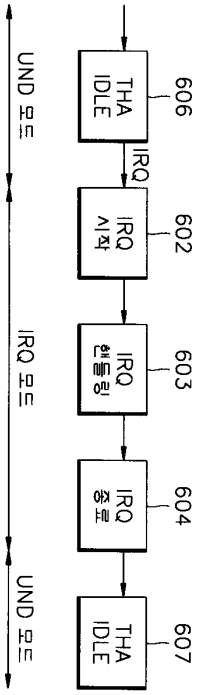
도면5


아이들 스레드
510
슈퍼 스레드
511
커널
502
인터럽트 핸들링
501
디바이스 드라이버
509
SCH_B
503
504
커널 접속 인터페이스부
스레드 THB1
506
스레드 THB2
507
스레드 ...
508
접속 인터페이스
505
FIQ/IRQ/SWI 모드
SVC 모드
USER 모드

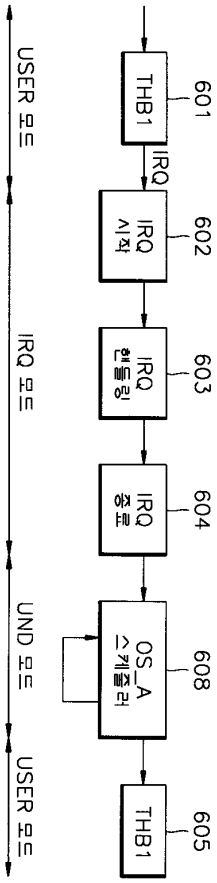
도면6a


USER 모드
THB1
601
IRQ
IRQ 시작
602
IRQ 핸들링
603
IRQ 종료
604
THB1
605
IRQ 모드
USER 모드

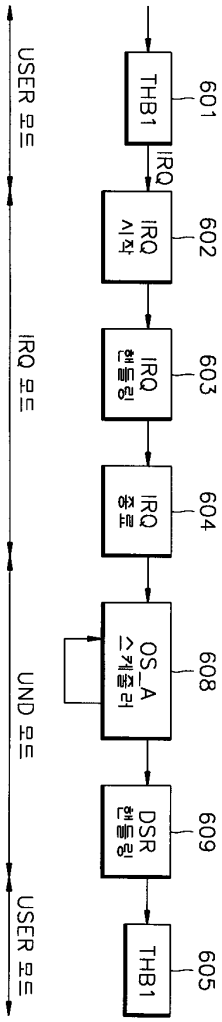
도면6b


THA
IDLE
606
IRQ
IRQ
시작
602
IRQ
핸들링
603
IRQ
종료
604
THA
IDLE
607
UND 모드
IRQ 모드
UND 모드

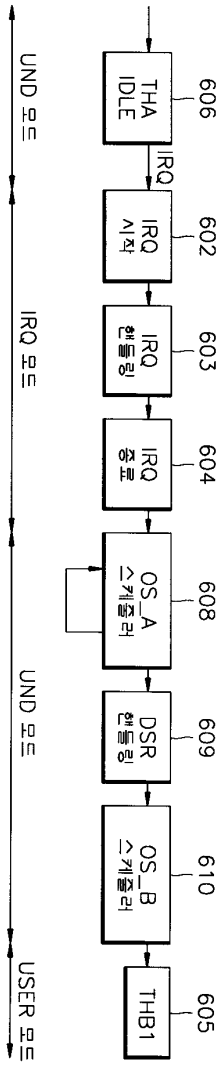
도면6c


THB1
601
IRQ
IRQ
시작
602
IRQ
핸들링
603
IRQ
종료
604
OS_A
스케줄러
608
THB1
605
USER 모드
IRQ 모드
UND 모드
USER 모드

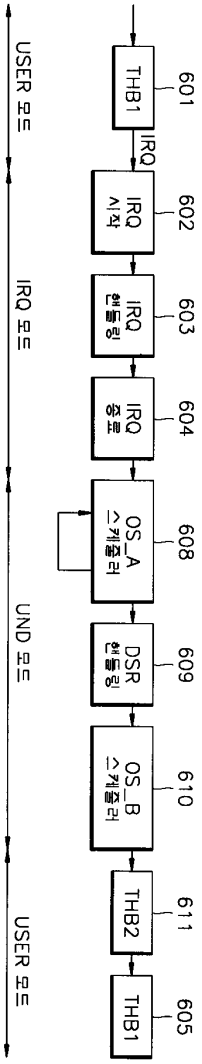
도면6d


USER 모드
IRQ 모드
UND 모드
USER 모드
THB1
601
IRQ
IRQ 시작
602
IRQ 핸들링
603
IRQ 종료
604
OS_A 스케줄러
608
DSR 핸들링
609
THB1
605

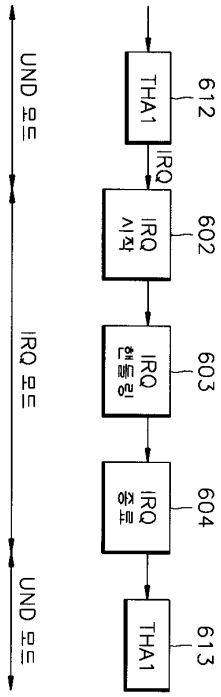
도면6e


THA
IDLE
606
IRQ
IRQ
시작
602
IRQ
핸들링
603
IRQ
종료
604
OS_A
스케줄러
608
DSR
핸들링
609
OS_B
스케줄러
610
THB1
605
UND 모드
IRQ 모드
UND 모드
USER 모드

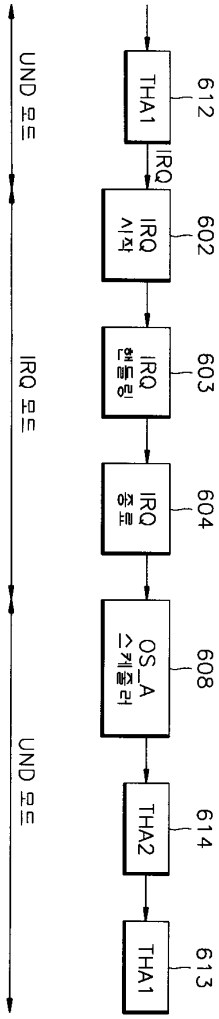
도면6f


USER 모드
IRQ 모드
UND 모드
USER 모드
THB1
601
IRQ
IRQ 시작
602
IRQ 핸들링
603
IRQ 종료
604
OS_A 스케줄러
608
DSR 핸들링
609
OS_B 스케줄러
610
THB2
611
THB1
605

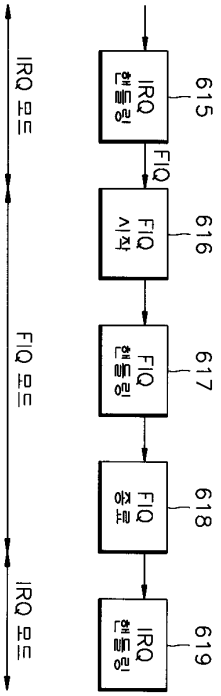
도면6g


UND 모드
IRQ 모드
UND 모드
THA1
612
IRQ
IRQ 시작
602
IRQ 핸들링
603
IRQ 종료
604
THA1
613

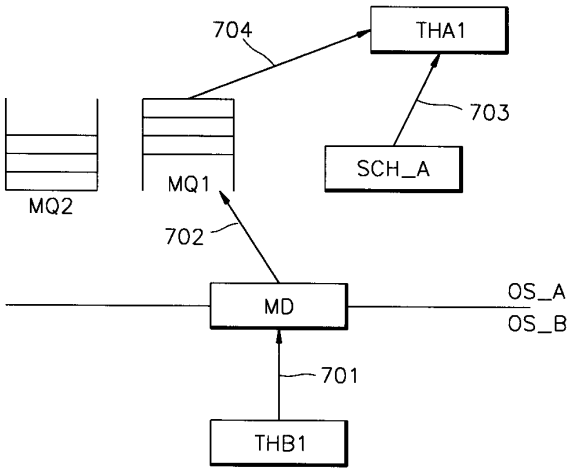
도면6h


UND 모드
IRQ 모드
UND 모드
THA1
612
IRQ
IRQ 시작
602
IRQ 핸들링
603
IRQ 종료
604
OS_A 스케줄러
608
THA2
614
THA1
613

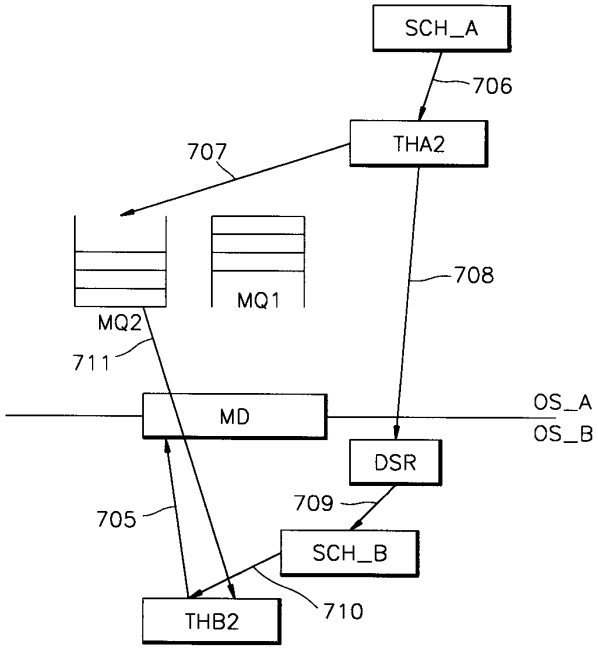
도면6i


IRQ 모드
FIQ 모드
IRQ 모드
IRQ 핸들링
615
FIQ
FIQ 시작
616
FIQ 핸들링
617
FIQ 종료
618
IRQ 핸들링
619

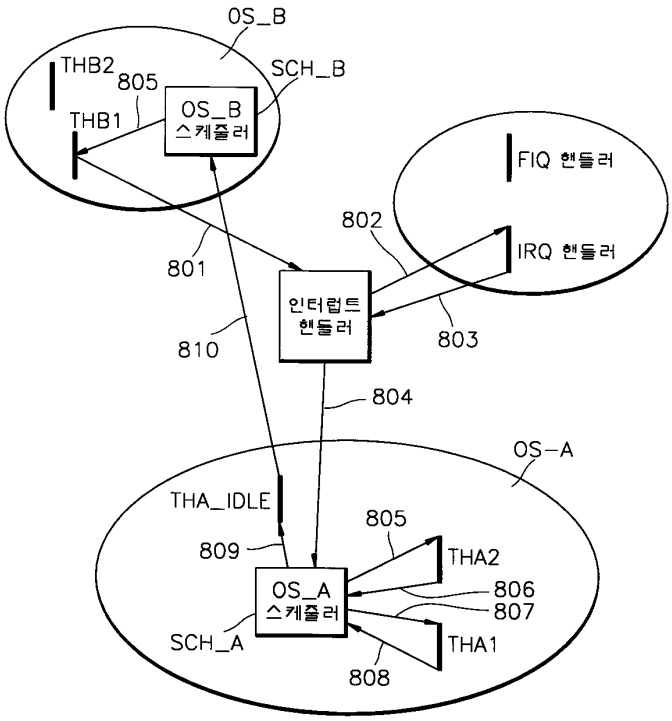
도면7a


704
THA1
703
SCH_A
MQ1
MQ2
702
MD
OS_A
OS_B
701
THB1

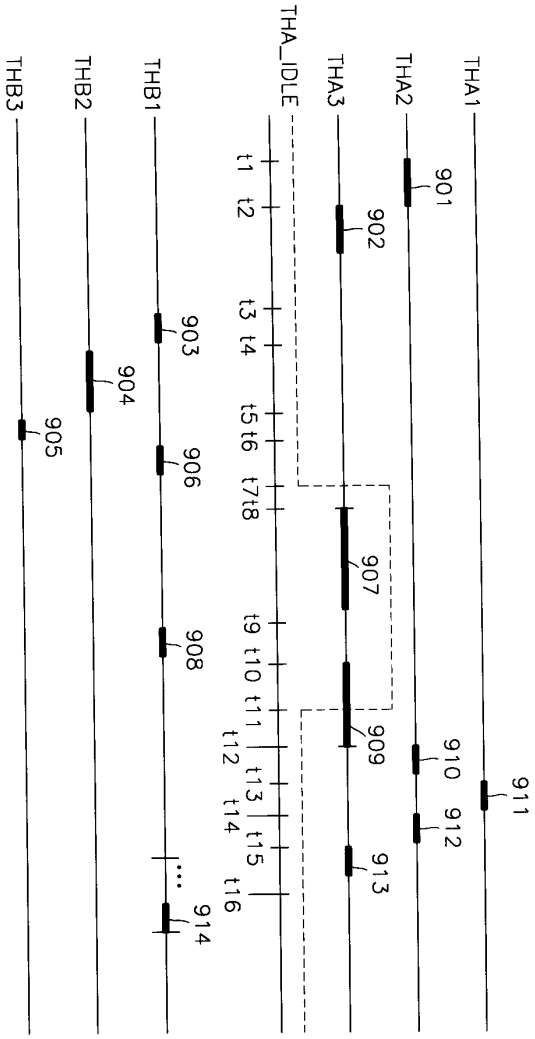
도면7b


SCH_A
706
THA2
707
708
MQ1
MQ2
711
MD
OS_A
OS_B
DSR
709
705
SCH_B
710
THB2

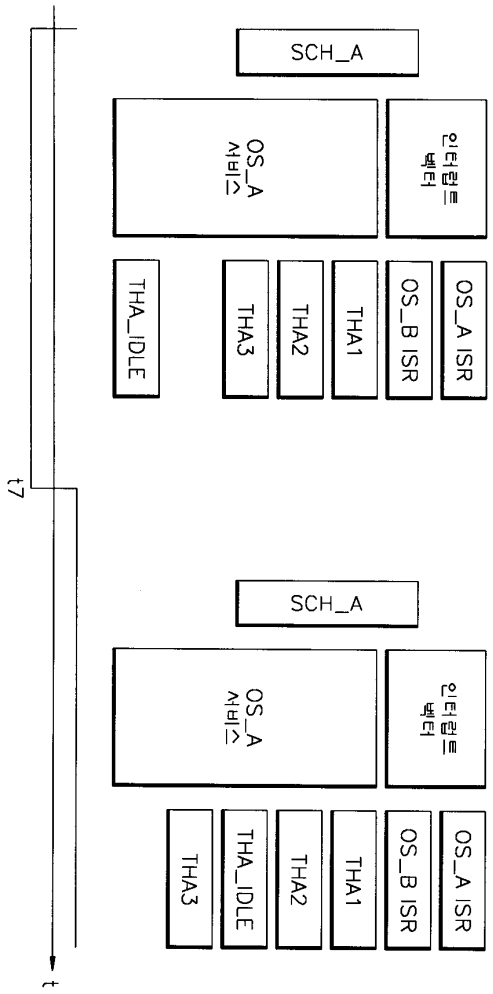
도면8


OS_B
THB2
805
SCH_B
THB1
OS_B
스케줄러
FIQ 핸들러
802
IRQ 핸들러
801
인터럽트
핸들러
803
810
804
OS-A
THA_IDLE
805
809
THA2
OS_A
스케줄러
806
807
SCH_A
THA1
808

도면9


THA1
THA2
THA3
THA_IDLE
THB1
THB2
THB3
901
902
903
904
905
906
907
908
909
910
911
912
913
914
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
t12
t13
t14
t15
t16

도면10


SCH_A
OS_A 서비스
인터럽트 벡터
OS_A ISR
OS_B ISR
THA1
THA2
THA3
THA_IDLE
t7
SCH_A
OS_A 서비스
인터럽트 벡터
OS_A ISR
OS_B ISR
THA1
THA2
THA_IDLE
THA3
t