범용 내장형 컴퓨터 시스템의 구현

장 위 식[†]·조 병 헌[†]·성 영 락[††]·오 하 령[††]

요 약

본 논문에서는 범용 내장형 컴퓨터 시스템을 설계하고 구현한다. 범용 내장형 컴퓨터 시스템의 가장 중요한 점은 확장성과 유연성이다. 개발된 시스템은 세 개의 모듈로 나뉘어 구성되며, 소프트웨어는 하드웨어에 독립적인 응용 프로그램 인터페이스를 제공한다. 또한 공개 실시간 운영체제인 uC/OS-Ⅱ의 기능을 확장하여 개발된 범용 내장형 컴퓨터 시스템에 이식한다. 확장된 uC/OS-Ⅱ는 여러 프로세스들이 같은 우선순위를 가지는 것을 지원하고, 같은 우선순위의 프로세스들에 대해서는 라운드-로빈 방식으로 스케줄링 된다.

Implementation of a general purpose embedded computer system

Wee Sik Jang[†]·Byeong Heon Cho[†]·Yeong Rak Sung[††]·Ha Ryoung Oh[††]

ABSTRACT

In this paper, a general purpose embedded computer system is designed and implemented. The most crucial points of the system are extensibility and flexibility. The hardware of the developed system is composed of three modules and the software provides hardware independent application program interfaces. Moreover, uC/OS-Ⅱ, a well-known open realtime kernel, is extended and ported onto the system. The extended uC/OS-Ⅱ supports that multiple processes can have the same priority and such processes are scheduled in a round-robin manner.

키워드 : 내장형 컴퓨터(Embedded System), 실시간 운영체제(Real-time Operating System)

1. 서 론

내장형 컴퓨터(Embedded Computer) 시스템은 특정한 작업만을 수행하는 것을 목적으로 하는 컴퓨터 시스템이다. 최근 들어, 입는 컴퓨터, 멀티미디어 PDA, 웹 기능이 내장된 가전제품 (TV, 냉장고), 자동차 등의 분야에서 내장형 컴퓨터의 사용분야가 크게 확대되고 있다. 하나의 내장형 컴퓨터 시스템을 개발한 후에 그 시스템을 다른 종류의 작업에 적용하려면 전체 시스템이 수정되어야 하는 경우가 많다. 이러한 비효율성을 줄이기 위해서는 하드웨어와 소프트웨어의 호환성 및 확장성을 높인 범용 내장형 컴퓨터 시스템의 개발이 필요하다. 범용 내장형 컴퓨터 시스템은 확장·변경이 용이하도록 하드웨어는 모듈화 되어야 하고, 소프트웨어는 계층구조로 구성되어야 한다. 또한 많은 경우에서 범용 내장형 컴퓨터 시스템에서는 동시에 다양한 작업이 수행되어야 하기 때문에 실시간 운영체제가 필요하다.

본 논문에서는 다양한 용도로 사용할 수 있는 범용 내장형 컴퓨터 시스템의 요구사항을 분석하고, 이를 바탕으로 범용 내장형 컴퓨터 시스템의 하드웨어와 소프트웨어를 설계하고 구현한다. 개발된 시스템의 하드웨어는 확장성을 고려하여 세 개의 모듈로 나뉘어 구성되며, 소프트웨어는 하드웨어에 독립적인 응용 프로그램 인터페이스를 제공한다. 또한 공개 실시간 운영체제인 uC/OS-Ⅱ[1]의 기능을 확장하여 개발된 범용 내장형 컴퓨터 시스템에 이식한다. 확장된 uC/OS-Ⅱ에서는 여러 프로세스들이 같은 우선순위를 가지는 것을 지원하고, 같은 우선순위의 프로세스들에 대해서는 라운드-로빈 방식으로 스케줄링 된다. 또한 프로그램 디버깅을 돕기 위하여 예외 처리 루틴들을 개발한다. 그래서, 예외상황이 발생하면 관련 정보들이 기록되어 발생된 예외상황에 대한 프로그램 오류를 쉽게 찾을 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구의 필요성 및 범용 내장형 컴퓨터 시스템이 필요로 하는 요구사항을 알아보고, 3장과 4장에서는 2장에서 알아본 요구사항에 맞

† 준 회 원 : 국민대학교 대학원 전자공학부
†† 정 회 원 : 국민대학교 전자공학부 교수
논문접수 : 2002년 7월 12일, 심사완료 : 2002년 10월 25일

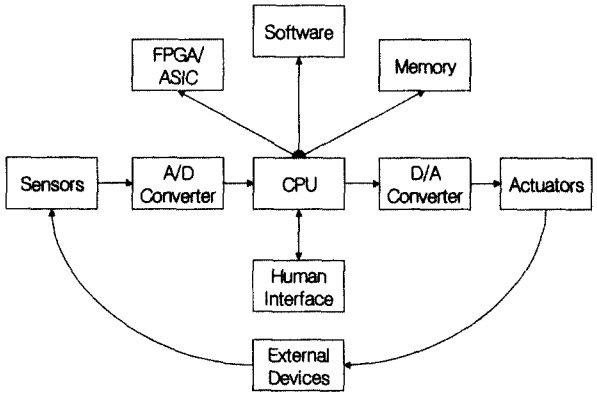
게 각각 하드웨어와 소프트웨어의 설계 및 구현을 다루고, 마지막으로 5장은 본 논문의 결론이다.

2. 범용 내장형 컴퓨터 시스템

본 절에서는 내장형 컴퓨터 시스템 개발의 난점들을 살펴보고, 범용 내장형 컴퓨터 시스템이 갖추어야 할 조건들을 분석한다.

2.1 내장형 컴퓨터 시스템 개발시의 난점

(그림 1)은 일반적인 내장형 컴퓨터 시스템의 구성이다[2]. 많은 내장형 컴퓨터 시스템들이 기본적으로 매우 유사한 구성을 가지지만, 사용되는 센서, 액츄에이터, 사용자 인터페이스 장치의 종류와 개수, 기타 시간적인 제약 사항들에 따라서 필요한 CPU의 성능, 메모리의 크기, A/D 및 D/A 변환기의 개수 등이 다르다.



(그림 1) 내장형 컴퓨터 시스템의 구성

이러한 내장형 컴퓨터 시스템의 개발하기 위해서는 여러 해결해야 할 난제들이 있다. 우선 첫째로 개발된 시스템의 동작을 검증하는 것이다. 일반적인 컴퓨팅 환경에서도 소프트웨어의 검증은 매우 어려운 문제이다. 그런데 내장형 컴퓨터 시스템 환경은 일반적인 환경에 비해서 디버거 등의 시스템 소프트웨어가 매우 열악하다. 또한 실제 하드웨어와 맞물려서 테스트해야 하므로 시스템의 동작을 검증하기가 매우 어렵다. 둘째는 시스템을 개발하는 데에 소요되는 비용이 크다. 앞서 언급한대로 내장형 컴퓨터 시스템은 개발 환경이 열악하기 때문에 개발에 소요되는 금전적, 시간적 비용이 크다. 셋째는 시스템의 사양이 바뀔 경우에 전체 시스템이 수정되어야 할 경우가 많다. 일반적인 컴퓨팅 환경의 경우와는 달리 내장형 컴퓨터 시스템의 개발을 시작하는 단계에서는 사양이 확정되지 못하는 경우가 많다. 그러므로 개발기간 중에 끊임없이 변경되는 사양을 지원하도록 설계가 변경되어야 한다. 이러한 현상은 개발이 종료된 이후에도 나타날 수 있다. 그러므로 앞서 난제들을 해결하였다고 하더라도 유사한 일들을 반복적으로 수행해야 하는 경우가 많다. 이러한 문제점들을 완화하기 위해서는 다양한 용도로 사용할 수 있는 범용 컴퓨터 시스템을 개발하고, 응용 분야나 사양의 변화에 대해서도 최소한의 변경만으로 사용할 수 있도록 해야 한다.

2.2 범용 내장형 컴퓨터 시스템의 요구 사항

범용 내장형 컴퓨터 시스템으로 이용되기 위해서는 하드웨어와 소프트웨어의 호환성 및 확장성이 중요하다. 그러므로 하드웨어는 모듈화 되어야 하고, 소프트웨어는 계층구조로 구성되어야 한다.

2.2.1 하드웨어

범용 내장형 컴퓨터 시스템의 하드웨어는 다음의 요구사항을 만족하여야 한다.

(R.1) 시스템의 일부분이 변경될 경우 다른 부분에 주는 영향을 최소화해야 한다.
(R.2) 다양한 프로세서들을 지원할 수 있어야 한다.
(R.3) 공통적으로 사용되는 입출력 장치들을 내장하여야 한다.

첫 번째는 시스템의 확장성에 대한 것이다. 본 논문에서는 범용 내장형 컴퓨터 시스템을 개발하여 다양한 용도로 사용하고자 한다. 그러나 제안된 범용 내장형 컴퓨터 시스템이 모든 내장형 컴퓨터 시스템의 대용품이 될 수는 없다. 만약 제안된 내장형 컴퓨터 시스템을 만능으로 사용하기 위해 절대 변경 사항이 발생하지 않는 시스템으로 만들고자 한다면 그 하드웨어의 비용은 무한대가 될 것이다. 그러므로 본 논문에서 개발하는 내장형 컴퓨터 시스템에서도 하드웨어의 변경이 있을 수 있다. 다만 변경될 가능성을 최소화하고 변경에 따른 비용을 줄일 수 있도록 하드웨어의 구조가 설계되어야 한다.

두 번째 요구사항은 시스템의 동작 속도에 대한 요구사항이다. 범용 내장형 컴퓨터 시스템은 다양한 작업에 사용될 수 있어야 한다. 작업 중에는 빠른 처리를 필요로 하는 경우도 있고, 상대적으로 느려도 큰 지장이 없는 경우도 있다. 전자의 경우에는 반드시 처리속도가 빠른 프로세서를 사용해야 하지만, 후자의 경우에는 저가의 프로세서를 사용하는 것이 경제적으로 이익이 된다. <표 1>은 범용 내장형 컴퓨터 시스템이 적용되는 분야와 필요한 하드웨어의 성능을 나타낸 것이다[2]. 표에서 나타난 바와 같이, 사용 목적

에 따라 필요한 프로세서의 속도, 입출력 속도 및 메모리 용량이 다르기 때문에 범용 내장형 컴퓨터 시스템은 다양한 프로세서를 지원하고, 입출력 장치 및 메모리 용량 등을 목적에 맞게 선택할 수 있어야 한다.

〈표 1〉 내장형 컴퓨터 시스템의 응용분야 및 하드웨어의 성능[2]

	Signal Processing	Critical Mission	Distributed	Small
Computing Speed (MIPS)	1000	10~100	1~10	0.1
I/O Transfer Rates (MB/sec)	1000	10	0.1	0.001
Memory Size (MB)	32~128	16~32	1~16	0.001

마지막은 시스템의 입출력에 대한 요구사항이다. 내장형 컴퓨터 시스템과 연결되는 외부 장치들은 매우 다양하다. 그러나 많은 장치들이 비슷한 전기적인 규격을 이용하므로 필요한 전기적인 규격의 수는 그리 많지 않다. 일반적으로 디지털 입·출력, 아날로그 입·출력, 직렬포트 등이 가장 일반적인 인터페이스 규격이다. 개발되는 범용 내장형 컴퓨터 시스템에서는 입출력 방법과 시스템에 부착되는 장치가 변경되더라도 전체 하드웨어가 변경되지 않도록 하는 것이 바람직하다. 그러므로 범용 내장형 컴퓨터 시스템에서는 일반적인 범용 내장형 컴퓨터 시스템에서 공통적으로 필요한 장치들을 내장하여 하드웨어가 변경될 가능성을 최소화해야 한다. 이 경우 내장되는 장치의 종류와 개수는 전체적인 하드웨어 시스템의 가격의 상승과 새로 개발할 경우의 소요되는 비용에 따라 적절하게 선택되어야 한다.

2.2.2 소프트웨어

범용 내장형 컴퓨터 시스템의 소프트웨어는 다음의 요구사항을 만족하여야 한다.

(R.4) 다양한 하드웨어를 지원하고 다중 태스크를 지원하는 실시간 운영체제가 이식되어야 한다.

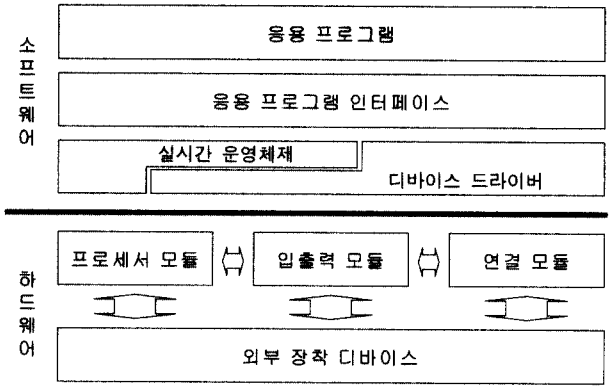
(R.5) 하드웨어가 변경되었을 경우에도 소프트웨어, 특히 사용자가 작성한 응용 소프트웨어에서의 변경이 최소화되어야 한다.

첫 번째 요구사항은 시스템의 운영체제에 관한 것이다. 범용 내장형 컴퓨터 시스템은 일반적으로 여러 가지의 작업을 동시에 처리해야 한다. 그러므로 여러 태스크들을 생성할 수 있고 그들이 병행하여 처리될 수 있는 운영체제가 필요하다. 그리고 대부분의 작업들이 시간적인 제약성을 가지고 있으므로 실시간 운영체제가 적합하다. 또한 (R.1)에 따르면 하드웨어적으로 다양한 프로세서가 지원될 수 있어야 하므로, 운영체제 또한 다양한 프로세서를 지원해야 한다.

두 번째는 응용 프로그램을 개발하는 환경에 대한 요구사항이다. (R.3)에서 지적한 대로 범용 내장형 시스템에서도 입출력 장치의 변경이 필요한 경우가 있다. 그러면 필연적으로 그 장치들을 제어하는 루틴의 변경이 필요하다. 만약에 이런 루틴들이 사용자 프로그램 안에 서로 실타래처럼 얽힌 채로 작성되어 있었다면 루틴을 변경을 위해서는 많은 시간과 노력이 필요할 것이다. 이러한 비용들을 줄이기 위해서는 확장과 변경이 용이하도록 소프트웨어 구조가 설계되어야 한다.

2.3 전체 시스템의 구성

앞서 제시된 (R.1)~(R.5)의 요구사항을 바탕으로 하여 범용 내장형 컴퓨터 시스템의 하드웨어와 소프트웨어가 설계되었다. (그림 2)는 설계된 시스템의 전체적인 구성이다. 하드웨어와 소프트웨어의 자세한 설계 및 구현에 관한 내용은 각각 3장과 4장에서 다룬다.



(그림 2) 범용 내장형 컴퓨터 시스템의 구조

3. 하드웨어

(그림 2)에는 2장에서 제시된 요구를 반영한 하드웨어의 구조를 제시하고 있다. 범용 내장형 컴퓨터 시스템의 하드웨어는 프로세서 모듈, 입출력 모듈, 연결 모듈의 3부분으로 구성된다. 각 모듈은 여러 외부 장치들과 연결된다.

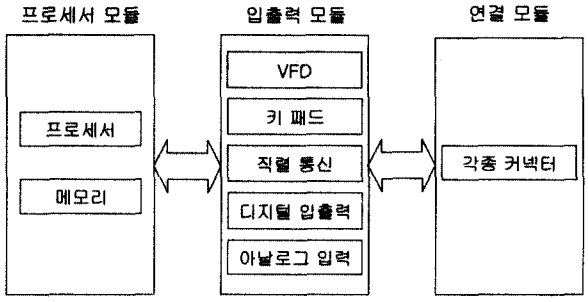
3.1 설 계

(R.1)~(R.3)의 요구사항을 만족하기 위해서는 기능 수정이 쉽도록 전체 하드웨어를 여러 모듈로 나누는 것이 바람직하다. 우선 (R.2)에 의해 프로세서가 변경될 경우에 전체 시스템에 주는 영향을 줄이기 위해서는 프로세서와 프로세

서의 영향을 크게 받는 부분들을 하나의 모듈로 독립시키고 시스템의 나머지 부분들과의 인터페이스를 정의하는 것이 필요하다. 한편 (R.3)에 의해 기본적인 입출력의 장치들을 지원한다 할지라도 입출력 핀들과 외부 장치를 연결하는 커넥터의 규격이 변경되어 새롭게 하드웨어를 설계해야 하는 경우가 많다. 이러한 재설계 비용을 줄이려면 입출력 장치들과 입출력 커넥터의 물리적인 연결 부분을 분리하여 각각을 독립적인 모듈로 설계하는 것이 필요하다. 이러한 점들을 종합하면 결론적으로 (그림 2)에 제시된 바와 같이 전체 하드웨어를 프로세서 모듈, 입출력 모듈 그리고 연결 모듈로 나누어 설계해야 한다. 이 경우, 프로세서를 교체할 경우에는 프로세서 모듈만을, 입출력 장치를 확충할 경우에는 입출력 모듈과 연결 모듈을, 외부장치들과의 커넥터 규격만 바뀔 경우에는 연결 모듈만을 수정하면 시스템의 다른 부분에 영향을 주지 않고 하드웨어를 변경할 수 있다. 이러한 특징은 개발 시간과 비용의 단축으로 귀결된다. 다만 확장성을 충분히 고려하여 프로세서 모듈과 입출력 모듈간의 인터페이스가 정의되어야 한다.

3.2 구 현

설계된 내용을 바탕으로 범용 내장형 컴퓨터 시스템을 프로세서 모듈, 입출력 모듈 그리고 연결 모듈로 나누어 구현하였다. 그림 3은 하드웨어의 구성을 나타낸다. 각 모듈 사이의 인터페이스를 살펴보면, 프로세서 모듈과 입출력 모듈 사이에는 프로세서의 데이터 버스, 읽기/쓰기 신호, 입출력 모듈에 장착되는 각종 장치들을 선택하기 위한 신호들이 포함된다. 또한 입출력 모듈과 연결 모듈 사이의 인터페이스는 입출력 모듈에서 외부로 연결되어야 하는 모든 신호들로 구성된다.



(그림 3) 하드웨어 구성도

프로세서 모듈은 AM188ES[3]와 MC68340[4]의 두 종류의 프로세서로 구현하여 사용 목적에 맞게 선택할 수 있도록 하였다. AM188ES는 인텔 계열의 16비트 프로세서이고, MC68340은 모토롤라의 32비트 프로세서이다. 각 프로세서 모듈은 리셋 회로, 클록 발생회로, 플래시메모리, 램, 와치독(watchdog) 타이머 등의 회로로 구성된다.

플래시메모리에는 시동 프로그램 및 주 프로그램이 저장된다. 본 논문에서는 시스템의 가격을 낮추면서도 신뢰도를 높이기 위하여, 플래시메모리가 시스템 파라미터를 저장하는 EEPROM의 역할도 겸하도록 하였다. 그러나 플래시메모리는 데이터를 쓰기 전에 반드시 지워야 하고 지우기는 블록 단위로만 가능하기 때문에 동작이 느린 단점이 있다. 이런 단점을 보완하기 위하여 본 논문에서는 플래시메모리

```
typedef enum {
    PARAM1 = 0,
    PARAM2,      /* 1 */
    PARAM3,      /* 2 */
    N_PARAMS,    /* 3 */
    END_OF_LOG = 0xFFFF
} PTYPE ;

typedef struct {
    PTYPE    type ;
    WORD     value ;
} PLOG;

WORD     param[N_PARAMS] ;
PLOG*    pblock = PB_ADDR ;

void ReadParams()
{
    while (pblock -> type != END_OF_LOG) {
        param[pblock -> type] = pblock -> value ;
        pblock++;
    }
}
```

	Offset	Content
PB_ADDR →	00H	0000H
	02H	0000H
	04H	0001H
	06H	0007H
	08H	0002H
	0AH	0044H
	0CH	0000H
	0EH	0055H
	10H	0000H
	12H	00F2H
	14H	0002H
	16H	00F4H
	18H	FFFFH

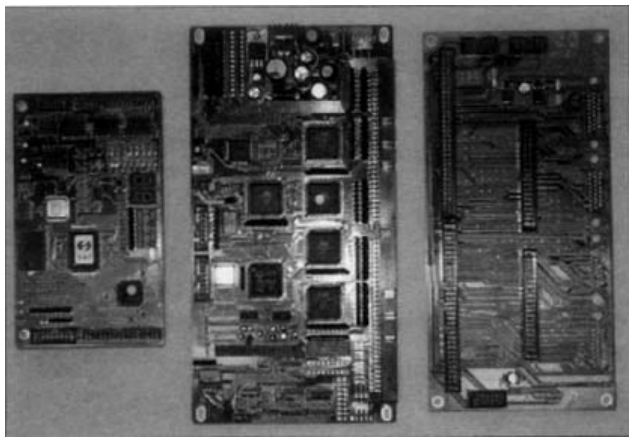
(그림 4) 파라미터 블록 관련 함수 및 파라미터 블록의 구조

의 일부분을 파라미터 블록으로 지정하고, 2개의 파라미터 블록을 파라미터들의 로그저장 구조로 번갈아 사용한다. (그림 4)는 본 논문에서 사용된 플래시메모리의 파라미터 블록에 관련된 함수와 파라미터 블록의 구조를 나타낸 것이다. 플래시메모리의 파라미터 블록은 PLOG 구조체의 어레이로 취급된다. 각 PLOG 구조체는 파라미터의 종류를 나타내는 type과, 값을 나타내는 value로 정의된다. (그림 4)에서는 간략하게 3종류의 파라미터만 사용하는 경우를 나타내었다. 시스템이 리셋되면 ReadParams()함수가 호출된다. 이 함수에서는 파라미터 블록의 PLOG 구조체의 처음부터 조사하면서, 각 PLOG 구조체의 type값에 해당하는 파라미터의 값을 value값으로 수정한다. 이 과정은 type값이 PLOG 구조체의 끝을 나타내는 END_OF_LOG값이 될 때까지 반복된다. 예를 들어 (그림 4)의 경우에는 param[PARAM0]은 0→55H→F2H로, param[PARAM1]은 7로, param[PARAM2]는 44H→F4H로 수정된다.

입출력 모듈은 화면 표시, 키 입력, 디지털 입출력, 직렬 통신, 아날로그 입력 회로 등으로 구성된다. 화면 표시를 위해서 VFD가 사용되었다. 일반적인 환경과는 달리 대부분의 내장형 컴퓨터 시스템에서는 VFD나 LCD가 표시 화면으로 사용되는 경우가 많다. 본 논문에서 사용된 VFD는 LCD와도 하드웨어/소프트웨어적으로 호환되는 것을 이용하였으므로 LCD로 손쉽게 대체될 수 있다. 키 입력으로는 일반적인 내장형 컴퓨터 시스템 환경에서 많이 사용하는 5개의 스위치를 가진 키패드를 사용했다. 직렬 통신은 최대 4개의 채널을 지원한다. 이중 2개는 AM188ES와 MC68340 프로세서에 탑재된 것이고 나머지는 Z8530[5]을 이용한 회로를 구현하였다. 각 채널은 간단한 스위치 조작만으로 TTL 수준 혹은 RS-232C 수준으로 선택해서 사용 가능하다. 디지털 입출력은 출력 4포트, 입력 3포트, 입력 혹은 출력으로 사용 가능한 3포트를 제공한다. 각 포트는 8개의 신호로 이루어지므로 디지털 입력 24~48개, 디지털 출력 32~56개를 제공한다. 그리고 아날로그 입력은 8 채널 8 비트의 아날로그-디지털 변환기 두 개를 사용하여 최대 16개의 입력을 받을 수 있다. 아울러 모든 장치에는 보호 회로를 부착하여 전기적인 충격이나 잘못된 신호에 대한 내구성을 높였다.

연결모듈은 입출력 모듈 및 외부 장치들과의 연결을 위한 커넥터들로 구성된다.

(그림 4)는 개발된 시스템의 모습이다. 좌측에서부터 프로세서 모듈, 입출력 모듈, 연결 모듈의 순서이다. 프로세서 모듈은 입출력 모듈의 자식보드로 연결되며, 입출력 모듈과 연결 모듈은 2개의 64핀 플랫케이블로 연결된다.



(그림 5) 개발된 시스템의 하드웨어

4. 소프트웨어

(그림 2)에는 2장에서 제시된 요구를 반영한 소프트웨어의 계층구조를 나타내고 있다. 범용 내장형 컴퓨터의 소프트웨어 시스템은 크게 디바이스 드라이버, 실시간 운영체제와 응용 프로그램 인터페이스, 응용 프로그램의 네 부분으로 구성된다.

4.1 설 계

2절에서 제시된 범용 내장형 컴퓨터 시스템의 소프트웨어에 대한 두 가지의 요구사항은 운영체제와 응용 프로그램 인터페이스에 대한 것으로서 서로 독립적이다. 우선 (R.4)는 운영체제에 대한 요구사항이다. 범용 내장형 시스템은 대부분 여러 가지 작업들이 동시에 수행되어야하며, 그 작업들은 일정 시간 내에 완료되어야 하는 특징을 갖고 있다. 그래서 본 시스템에서는 실시간 운영체제를 사용한다. 현재 국내에서 많이 쓰이는 실시간 운영체제로는 마이크로텍의 VRTX[6], WindRiver의 pSOS[7], VxWorks[8] 등의 상용 제품이 있고, 그 외에도 간단하면서도 널리 쓰이는 uC/OS-II[1]나 이미 그 안정성을 확보 받은 RTEMS [9], GNU 컴파일러나 그 밖의 도구들을 지원하며 내장형 컴퓨터 시스템 시장에서 널리 알려진 시그너스의 eCos[10] 등의 공개 실시간 운영체제가 있다. 상용 제품의 경우 라이선스 가격이 매우 비싸므로, 본 논문에서는 공개 실시간 운영체제이면서 다양한 프로세서를 지원하고 수정이 편리한 uC/OS-II의 기능을 확장시켜 사용한다.

(R.5)의 요구사항을 만족하기 위해서는 하드웨어를 제어하는 소프트웨어를 계층 구조로 만들어야 한다. 즉 하드웨어를 직접 제어하고 하드웨어에 종속적인 장치 드라이버 계층과 장치 드라이버 계층의 함수들을 이용하며 하드웨어에 독립적인 응용 프로그램 인터페이스 계층으로 나누는

것이 필요하다. 이 경우 사용자는 응용 프로그램 인터페이스를 만을 이용하여 프로그램을 작성한다면 하드웨어들이 변경되더라도 디바이스 드라이버의 재작성만으로 응용 프로그램이 동작할 수 있다. 디바이스 드라이버 중 입출력에 관계되는 함수들의 원형은 ANSI C 표준 라이브러리의 형태를 유지되도록 개발한다. 또한 응용 프로그램 인터페이스에서는 응용 프로그램을 작성하는데 필요한 각종 함수들의 프로토타입을 정의하여 운영체제와 디바이스 드라이버의 내용이 변경되더라도 응용 프로그램의 변경을 최소화하도록 한다.

4.2 구 현

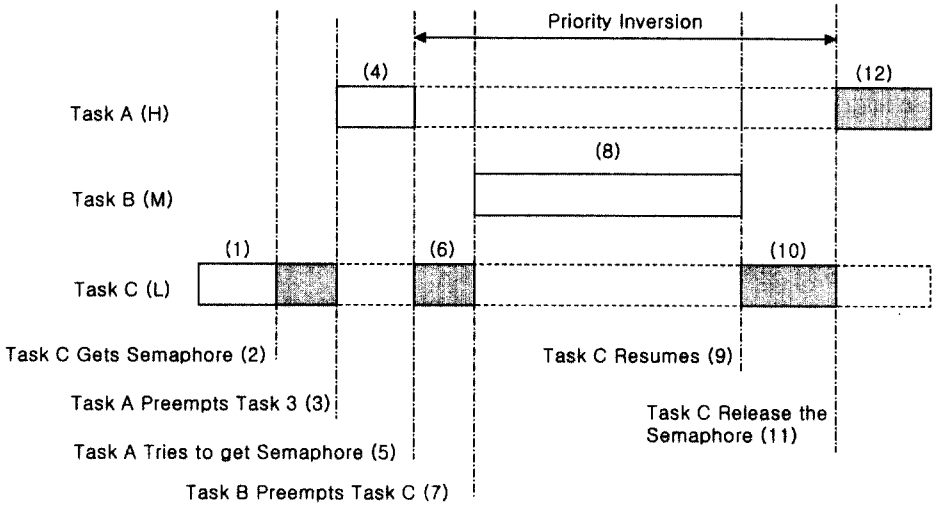
4.2.1 실시간 운영체제

본 시스템의 운영체제인 uC/OS-Ⅱ는 크기가 작고, 다양한 프로세서를 지원하며, 태스크를 지원하기 때문에 프로그램 작성이 쉽다. 또한 그 구조가 간단해서 빠르게 동작하고, 작은 규모의 응용 프로그램 개발에 적합하다. 또한 실시간 처리를 위하여 모든 태스크에는 우선순위가 부여되며, 우선순위 기반으로 태스크들을 스케줄 한다[1].

그러나 공개된 uC/OS-Ⅱ는 프로세스의 개수가 64개로 제한되어 있어 많은 태스크를 요구하는 시스템에는 적용이 불가능하고, 또한 우선순위 역전[1]의 문제가 있다. 우선순위 역전이란 우선 순위가 낮은 태스크가 우선 순위가 높은 태스크 보다 먼저 수행되는 것을 말하는 것이다. 예를 들어, (그림 5)의 경우에서 태스크 A와 태스크 C는 같은 자원을 사용하는 태스크라고 하자. 이런 상황에서 태스크 C가 그 공유자원을 얻어서 수행하는 도중에 태스크 A와 B가 준비 상태가 되었다고 하자. 그러면 당연히 우선순위가 높은 태스크 A가 스케줄 될 것이다. 그러나 공유자원이 태스크 C에 의해 사용 중이므로 곧 대기 상태가 될 것이다. 이런 상황이 되면 우선순위가 낮은 태스크 우선순위가 높은 태스크 A보다 먼저 수행되는 현상이 발생되는데, 이를 우선순위 역전이라고 한다. 이 문제를 해결하려면 태스크 A가 대기 상태로 될 때, 한시적으로 태스크 C의 우선순위를 태스크 A와 같게 설정해 주면 된다. 그러나 현재의 uC/OS-Ⅱ에서는 모든 태스크들이 서로 다른 우선순위를 가져야 하므로 체계적인 해결이 어렵다.

이 두 가지 문제를 해결하기 위해서 본 논문에서는 여러 태스크들이 같은 우선순위를 갖는 것을 허용하고, 이 태스크들에 대해서는 라운드-로빈 스케줄링[11]을 하도록 변경하였다. 라운드-로빈 스케줄링은 시분할 시스템에서 사용하는 스케줄링 방법으로 일정 단위 시간마다 문맥교환을 통해 생성된 모든 태스크가 동일하게 수행되도록 하는 스케줄링 방식이다. 제안된 스케줄링 알고리즘을 구현하기 위해 본 논문에서는 TCB(task control block)와 스케줄러를 수정하고 같은 우선순위를 가지는 태스크들을 관리하기 위한 자료구조를 추가하였다[12].

한편 내장형 컴퓨터 시스템에서는 일반적인 경우에 비해서 소프트웨어 개발이 어렵다. 이것은 디버거를 포함하여 여러 프로그래밍 환경이 매우 열악하기 때문이다. 특히 프로세서에서 예외상황이 발생했을 경우에 대한 적절한 루틴이 없을 경우, 프로그램 개발이 체계적으로 진행되기 어렵다. 본 논문에서는 이러한 불편을 줄이기 위해서 예외상황 처리에 대한 루틴들을 개발하였다. 즉, 예외상황이 발생되면 발생한 예외상황의 종류, 위치, 리셋된 횟수 등을 플래시메모리의 파라미터 블록에 저장하고, 시스템을 재부팅하



(그림 6) 우선순위역전[1]

도록 한다. 그러면, 재부팅된 이후에 플래시메모리를 참조하여 예외상황을 일으킨 위치와 종류를 파악하여 프로그램을 수정할 수 있다.

4.2.2 응용 프로그램 인터페이스 및 장치 드라이버

응용 프로그램에서는 프로세서에 상관없이 동일한 방식으로 접근할 수 있도록 해야 하기 때문에 응용 프로그램 인터페이스는 하위 계층의 장치 드라이버와 운영체제를 이용하여 하드웨어, 특히 프로세서에 독립적으로 작성한다. 예를 들어 현재 운영체제의 상태를 VFD에 표시하고자 할 때는 void OSStatusDisplay(int value)라는 함수를 이용한다. 이 함수는 value 값에 따라 지정된 장치의 장치 드라이버를 이용하여 운영체제의 상태를 표시한다. 이 함수를 이용하면 새로운 출력장치로 출력할 경우에도 그 장치에 대한 장치 드라이버만 새로 작성하면 된다.

장치 드라이버는 물론 장치 별로 따로 작성되어야 하지만 사용되는 프로세서의 종류에 따라도 따로 드라이버를 작성해야 한다. 장치 드라이버가 프로세서에 따라 바뀌는 것은 각 프로세서에 따라 메모리 맵이 다르고, 액세스하는 방식이 다르기 때문이다. 장치 드라이버 중에서 운영체제에 종속적인 부분들은 모든 내용은 uC/OS-Ⅱ의 시스템 호출 함수들을 기반으로 작성했다. 또한 각각의 장치 드라이버는 다른 장치와 독립적으로 동작할 수 있도록 구현하였다. 기본적으로 작성한 디바이스 드라이버는 직렬 통신, 메뉴 처리, 메모리 관리, 아날로그-디지털 변환에 대한 라이브러리 등이다. 직렬 통신의 경우, 프로세서에서 탑재된 직렬 포트를 위한 장치 라이브러리와, Z8530의 직렬 포트를 위한 장치 드라이버가 존재한다. 또한 프로세서의 직렬 포트 중에 하나를 디버깅 전용으로 만들어 PC를 통한 프로그램 다운로드 및 디버깅 작업이 가능하도록 하였다.

5. 결 론

본 논문에서는 범용 내장형 컴퓨터 시스템의 요구사항을 분석하고 이를 바탕으로 하드웨어와 소프트웨어를 설계·구현하였다. 범용 내장형 컴퓨터의 하드웨어는 프로세서의 성능, 입출력 장치 등에 대한 확장성이 우선적으로 고려되어야 하며, 하드웨어의 변경시에 소요되는 비용을 최소화할 수 있어야 한다. 그래서 제안된 시스템의 하드웨어는 프로세서 모듈, 입출력 모듈, 연결 모듈로 나누어 구성하였다. 한편, 소프트웨어는 하드웨어 제어 루틴들을 응용 프로그램 인터페이스와 장치 드라이버로 계층적인 구조로 구성하였다. 이러한 계층적인 구조는 하드웨어의 변경에 따른 소프트웨어의 변경을 최소화하는 역할을 한다. 또한 시스템의 운영체제로 uC/OS-Ⅱ의 기능을 확장하여 이식하였다. 확장된 uC/OS-Ⅱ에서는 기존 uC/OS-Ⅱ에서 가지는 프로세스의 개수 제한 및 우선순위 기반의 스케줄링만을 제공하는 단점을 보완하였다. 그래서 같은 우선순위를 갖는 프로세스들을 지원하고 이들 사이에서는 라운드-로빈 방식으로 스케줄링 되도록 하였다.

참 고 문 헌

[1] Jean J. Labrosse, MicroC/OS-II The Real-Time Kernel, R & D Books, 1999.

[2] Philip J. Koopman, Jr, "Embedded System Design Issues," Proceedings of the International Conference on Computer Design, 1996.

[3] AMD, AM186ES and AM188ES Users Manual, 1997.

[4] Motorola, Motorola MC68340 Integrated Processor with DMA Users Manual, 1992.

[5] Zilog, Z8030/Z8530 Z-BUS® SCC/SCC Serial Communications Controllers, 1997.

[6] Mentor Graphics, VRTX-The Operating System for System-on-Chip, 1999.

[7] WindRiver Systems, pSOSystems 2.5, 2000.

[8] WindRiver Systems, VxWorks AE, 2001.

[9] http://www.rtems.com.

[10] Red Hat, eCos Embedded Configurable Operating Systeme, 2001.

[11] Abraham Silberschatz, Peter B. Galvin, Operating System Concepts, Addison Wesley, 1998.

[12] 장위식, 오하령, 성영락, "라운드-로빈 스케줄링을 지원하는 uC/OS Ⅱ의 구현", 제1회 정보가전과 실시간 시스템 응용 워크숍, 2001.

장 위 식

e-mail : amugeona@elco.kookmmin.ac.kr

2000년 국민대학교 전자공학과(공학사)

2002년 국민대학교 전자공학과(공학석사)

관심분야 : 내장형 컴퓨터, 실시간 운영체제

조 병 헌

e-mail : d995552@hanmail.net

1997년 국민대학교 전자공학과(공학사)

1999년 국민대학교 전자공학과(공학석사)

1999년~현재 국민대학교 전자공학과 박사과정

관심분야 : 내장형 컴퓨터, 실시간 운영체제

성 영 락

e-mail : yeong@kookmin.ac.kr
1989년 한양대학교 전자공학과(공학사)
1991년 한국과학기술원 전기및전자공학과 (공학석사)
1995년 한국과학기술원 전기및전자공학과 (공학박사)
1995년~1996년 한국과학기술원 위촉연구원
1996년~1998년 국민대학교 전임강사
1998년~2002 국민대학교 조교수
2002년~현재 국민대학교 전자공학부 부교수
관심분야 : 멀티미디어 시스템, 시스템 모델링 및 시뮬레이션, 병렬처리

오 하 령

e-mail : hroh@kookmin.ac.kr
1983년 서울대학교 전기공학과(공학사)
1983년~1986년 삼성전자 종합연구소
1988년 한국과학기술원 전기전자과 컴퓨터 공학전공(공학석사)
1992년 한국과학기술원 전기전자과 컴퓨터 공학전공(공학박사)
1992년~1996년 국민대학교 공과대학 전자공학부 조교수
1996년~2001년 국민대학교 공과대학 전자공학부 부교수
2001년~현재 국민대학교 공과대학 전자공학부 교수
관심분야 : 컴퓨터구조, 운영체제, 분산 및 병렬처리, 멀티미디어