논문 02-01-06

통합설계 방식을 이용한 컨트롤 보드의 인터페이스 자동화 시스템

An Interface Automatic System on the Control Board using Hardware/Software Co-Design

印 致 虎*
Chi-Ho Lin*

요 약

본 논문은 8051 마이크로프로세서의 내부 코아 특성과 시스템 재사용에 대한 통합설계 방법을 사용하여 하나의 시스템을 제작하였다. 또한 이 시스템을 독립적으로 사용할 뿐만 아니라 다른 시스템의 모듈로서 사용할 수 있도록 시스템을 설계 및 구현한다.

제안된 방법에서 재사용이 가능하도록 시스템 자체를 객체형으로 구현하고, 시스템들 간의 연결을 위해 객체형이 구현된다. 이러한 객체형의 요구에 맞추어 시스템들이 자기정보를 가지고 다른 시스템들과 연결되었을 때 자신의 정보를 제공함으로써 자동 인식되고, 시스템 자체가 다른 시스템에 재 적용될 수 있도록 한다.

본 논문에서 제안된 방법은 기존의 Z-80 계열의 교육용 제어보드와 비교 분석하여, 제안한 시스템이 확장성의 효율성을 제공할 뿐만 아니라, 대부분의 기능을 소프트웨어로 처리하여 개발 기간, 비용 및 보드 크기가 축소되는 등의 장점을 입증하였다. 또한 객체형 시스템 아키텍처로 설계하여 확장성과 이식성이 증대되는 특징을 보였다.

Abstract

This paper manufacturing one system and use this separatively, plan, and embody system that apply integration design method in research about characteristic of internal core of 8051 micro-processors and system reusability so that can use as module of other system.

The proposed system itself by object style so that reusability may be possible in proposed method and object style for connection between this systems is required. Set on these request and when systems have own information and were linked with other systems, by supplying own information automatic movement itself is realized and system itself embodies ashes so that can be applied to other system.

The proposed method in this paper analyzes and compares with existent Z-80 education board, as well as system that propose offers extensibility, it handles most function to software and development period, expense and baud dimension confirmed advantage of and so on that reduce. Also, design for object style system architecture and showed feature that extensibility and portability are augmented.

Keyword : Co-Design, Embedded System, Object System, PNP, Control Board

* 世明大學校 컴퓨터 科學科
(Dept. of. Computer Science, Semyung Univ.)

接受日:2002年 4月 25日, 修正完了日:2002年 7月 28日

I. 서 론

최근 주문형 반도체 기술이 발달함에 따라 국내외에 많은 반도체 회사들이 설립되고 이들은 앞 다투어 저렴한 가격과 높은 성능을 가진 제품들을 내놓고 있다. 이와 함께 발전되어 온 것이 보드 설계기술이다. 설계기술은 보드의 크기는 줄이면서도 성능은 더욱 향상되어지는 형태로 발전되어 지고 있다. 제어용 인쇄기판 설계는 산업용의 주문형 제어보드 개발에 치중해 온 것이 사실이다. 또한 기존의 설계 방법으로는 매우 복잡해지고 여러 가지 기능을 수행하는 시스템을 설계하는데 많은 어려움이 발생되는 것이 기존의 현실이다. 이러한 시장의 추세와 설계 흐름에 따라 최적화 된 설계의 방법론적 요구와 실제 적용의 가능성이 요구되어졌다.[1]

이러한 요구에 부응하여 시스템을 소프트웨어와 하드웨어의 통합설계 방법이 나타나게 되었다. 통합설계 방법은 설계비용의 감소 및 전력 소비의 최소화, 회로보드의 크기를 줄일 수 있는 등의 장점을 가진다. 또한 통합설계 방법을 통한 시스템 설계상의 장점으로는 하드웨어(hardware)의 고성능과 소프트웨어(software)의 효율성을 이용한 설계방법이라는 점이다.[2][9]

본 논문에서는 8-bit 마이크로프로세서인 8051에 대한 각종 연구를 바탕으로 입출력 시스템의 구현을 통해 하나의 객체형 시스템 아키텍처를 구현한다. 본 논문에서는 이러한 이론적 배경을 정확하게 수용할 수 있는 시스템에 대하여 연구하고 이를 실제적으로 구현해보면서 이를 통한 효율성의 검증과 유용성에 대해 제시한다. 또한 가장 기본적인 제어보드를 구현하고 그 제어보드 상에서 PNP(Plug aNd Play) 기능을 삽입하여 모듈보드를 자동으로 인식할 수 있는 시스템 소프트웨어를 설계하고 설계된 자료와 임의의 규약을 지정하여 포팅하도록 하는 방법으로 구현하였다. 또한 여러 가지 제약조건과 테스트를 위한 특정 요인들에 대한 항목들을 먼저 명시하고 구현에 들어간다.[3][8]

II. 시스템의 설계 방법과 구성

2.1 통합설계 방법의 분류 및 적용

통합설계 방법은 서론에서도 언급한 바 있듯이 하드웨어와 소프트웨어의 구현상 장점을 절충하여 제안된 설계 방식이다. 하드웨어-소프트웨어 통합설계는 상위단계에서 기술된 고성능의 복잡한 시스템을 하드웨어 부분과 소프트웨어 부분으로 적절히 혼합하여 최적의 성능을 가지도록 설계하는 기술이다.

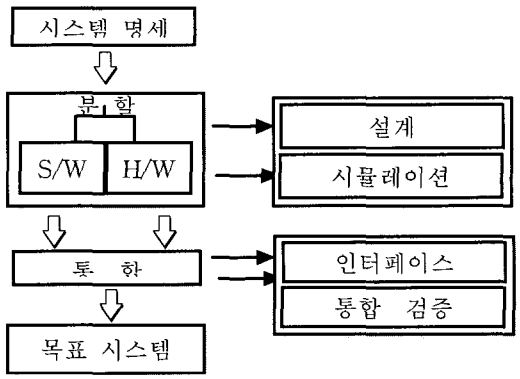

그림 1. 통합설계 방법에 따른 작업 흐름도
Fig. 1 Work flow by Co-Design Method

이러한 통합설계는 시스템의 성능 향상 및 설계 비용의 최소화 측면에서 매우 유용한 기술로 현재 활발한 연구가 진행되고 있다. 본 논문에서는 구현하는 과정을 보이기에 앞서 기본 설계 방법론으로 채택한 통합설계를 관점에 따라 세 가지 방법으로 분류하였다. 첫 번째는 순수 하드웨어 입장으로서 하드웨어 자체 즉, 칩 레벨에서 프로세서 혹은 DSP와 같은 복잡한 칩들을 구현하기 위해 하드웨어와 소프트웨어를 적절히 분할하여 설계하고 이를 하나의 칩으로 통합하여 구현하는 것이다.[2] 두 번째는 시스템 레벨에서 소프트웨어 와 하드웨어를 구분하여 설계 및 구현하는 방법이다. 이것은 하드웨어와 소프트웨어의 구분이 첫 번째의 경우보다 훨씬 정확하게 분할될 수 있다. 쉽게 보면 사용되어질 장치와 이를 구동시킬 수 있는 프로그램으로 구분되어진다고 할 수 있다. 하지만 이 또한 시스템의 일부 즉, 일종의 하드웨어라고 볼 수 있다는

사실이다. 즉, 프로그램 되어진 후에 하드웨어로서 장착되어지는 것을 말한다. 마지막 세 번째로는 시스템과 사용자 인터페이스를 구현하는 방법으로서 이는 일반적으로 PC의 윈도우 구동 드라이버와 이를 이용한 어플리케이션 프로그램까지 모든 것을 프로그램화하고 설계된 제어보드를 구동시키게 된다.

위의 세 가지 방법들은 모두 동일한 작업 흐름을 가진다. 즉, 그림 1과 같이 기존의 통합설계 방법이 그대로 적용된다. 첫 번째로 설계자의 생각을 시스템 사양으로 설계하게 되며, 두 번째는 설계사양을 하드웨어와 소프트웨어로 분할하고, 세 번째 하드웨어와 소프트웨어 간에 데이터를 공유시키고 통신할 수 있도록 통합하고, 설정하는 단계가 필요하다. 이러한 단계들을 거쳐서 마지막으로 완성된 통합된 시스템이 구현되는 것이다.

2.2 객체형 시스템 아키텍처의 구성

본 논문에서 제안하는 객체형 시스템 아키텍처란 소프트웨어의 개발형식에 사용되는 클래스 형태의 객체 타입을 하드웨어 시스템에 적용하여 구현되도록 한다는 것이다. 이를 통해 하나의 완성된 시스템은 다시 다른 시스템의 보조 시스템으로 재사용 될 수 있도록 한다는 것이며, 특정한 규약에 따라 상당한 호환성과 이식성 그리고 확장성을 가지도록 한다는 것을 그 목표로 하고 있다. 객체형 시스템 아키텍처를 구현하기 위해서는 몇 가지 제약사항들이 필요하다. 첫 번째 하나의 프로세서를 가져야만 한다는 것이다. 이것은 시스템이 자체적으로 데이터 처리 능력을 가져야 한다는 것이다. 이는 자신의 시스템 자체를 구동시키는 것을 기본 작업으로 하고, 외부로부터 들어오는 명령데이터와 정보데이터를 전송 받아 처리하는 것까지 수행하는 것을 말한다. 두번째 모든 시스템이 동일한 규약을 사용하도록 한다는 것은 앞에서 언급한 바 있듯이 시스템 아키텍처 내부에 서로간의 통신을 위한 특정 규약이 필요하다는 것이다. 특히 물리계층과 전송계층에 대한 중점적인 정의가 필요하며, 이를 정의함으로써 시스템들 간의 통신을 가능하도록 한다. 그 외에 각종 규약의 요소들은 인터럽트를 이용하게 된다. 따라서 이러한 시스템 구성에서 무엇보다 중요하게 고려되어야 할 부분이 커넥터의 크기와 전달요소가 된다.

III. 객체형 시스템의 설계 및 구현

본 제어보드는 크게 두 가지 부분으로 나누어 설계하였다. 즉, 메인보드와 모듈보드로 구분하며, 메인보드의 경우 하나의 프로세서와 롬, 램, 입출력을 위한 주변 장치로 구성하고, 모듈보드의 경우 객체형의 한 모델이 될 수 있는 모듈로서 하나의 프로세서와 모듈의 기능에 해당하는 장치들로 구성하게 된다. 본 논문에서는 모듈보드의 예로써 디스플레이(display)장치를 설계하였다.

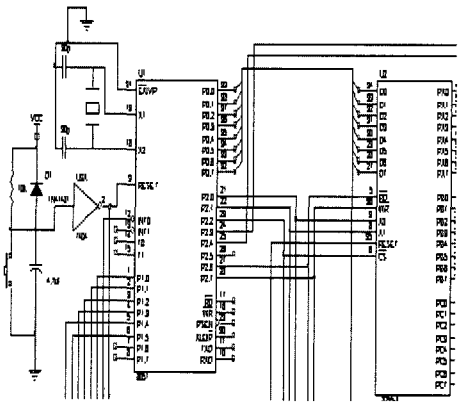

그림 2. 시스템의 설계 회로도
Fig. 2. A Circuit Diagram of System

그림 2.는 본 논문에서 제안한 시스템의 설계를 보여주고 있다. 그림 2와 같이 구성이 매우 단순하고 데이터의 흐름이 복잡하지 않다는데 장점이 있고, 따라서 개발 기간이 길 필요성이 없다는 것이다. 여기서는 필요한 것이 소프트웨어적인 처리가 어떻게 이루어져야 다른 모듈을 받아들이고 인식하는가에 대한 것이다.

본 논문에서 제안하는 시스템은 중앙 처리 장치와 주변 장치 사이에서 데이터의 통신과 이를 위한 설계와 배치, 그리고 설계에 필요한 프로그램 툴들에 대한 실습이 가능하고, 직접 제작할 수 있도록 쉽게 구현하는 것을 목적으로 하였다.

3.1 하드웨어의 설계 및 구현

먼저 메인보드에서 각 포트에 폴링(polling)형식으로 지속적인 신호를 보낸다. 이때 디스플레이보드가 연결되면 연결신호를 받게 되고 연결되지 않으면 내부 타이머에 의해 자동으로 다음 포트에 신호를 보내게 된다. 만약 연결이 되었다면 디스플레이 보드에 인터럽트를 걸게 되는데 이때 디스플레이 보드에서는 신호를 분석하여 일반 데이터가 아님을 확인하면 메인으로 연결되었다는 신호를 보내게 된다. 신호를 받은 메인보드의 CPU는 데이터를 보내고 디스플레이 작업을 수행하게되는 것이다. 본 제어보드의 경우 MPU로는 89C55를 사용하고, 2개의 8255를 사용하여 메인보드를 구성하였다. 모듈보드에는 89C2051 MPU를 사용하여 메인보드의 작업노드를 줄임과 동시에 모듈보드 자체의 정보를 가지고 있도록 설계하였다. 메인보드의 구성은 보드 설계상의 가장 기본적인 구성만을 가지고 설계하였으며, 나머지 부분들은 모듈보드에 별도로 설계하여 기능을 확장하는 개념으로 설계를 하였다. 이것은 보드 사이즈의 최소화함과 동시에 필요한 기능만을 확장하여 사용할 수 있는 구조가 필요하였기 때문이다. 모듈보드는 정형화된 포트의 구조를 가질 필요가 있기 때문에 포트의 구조를 동일하게 설계하였다. 그림 3은 실제 제어보드의 회로도이다.

그림 2에서 중앙을 기준으로 왼쪽의 메인보드 부분과 오른쪽의 모듈부분으로 나누어진다. 본 시스템은 메인보드의 경우 순수입출력만을 하므로 모듈 시스템을 호출하는 목적으로 축소하였다. 하지만 인터럽트를 통한 모듈시스템 서비스의 준비를 갖추어 두어야한다. 즉, 외부와 연결되는 부분만을 고려하는 순수 메인보드의 역할을 수행하도록 하였다.

3.2 소프트웨어의 설계 및 구현

본 논문에서 제안한 객체형 시스템 아키텍처를 설계하기 위해서 PNP 기능을 고려하여 설계한다. PNP 기능을 구현하기 위해서는 몇 가지 요소들을 갖추어야 한다. 즉, 새롭게 추가되는 보드는 자신에 대한 정보를 가지고 있어야 하고, 이 보드에 대한 영역이 메인보드에서 제공되어야 한다.

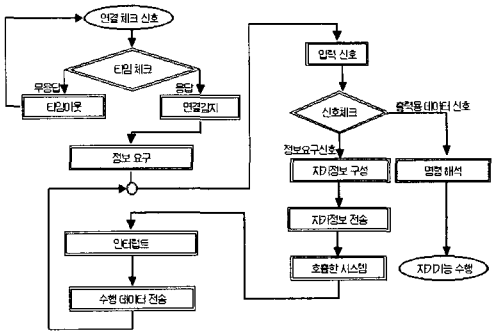

그림 3. 프로그램 상에서 신호의 흐름
Fig. 3. A stream of signal on the program

또한 새롭게 추가되는 보드는 자신이 받아야할 서비스와 자원의 할당에 대한 명세가 나타나 있어야 하고, 이를 받아들이는 메인보드는 이들의 요구에 대하여 정확한 서비스를 해야 한다. 먼저 외부에 접속을 감지하기 위해 폴링 루틴을 적용하여 체크 신호를 각각의 포트에 지속적으로 보낸다. 신호가 나가는 것과 동시에 타이머 인터럽트가 동작되도록 하였으므로 만약 특정한 시간동안 아무런 응답이 없으면 타임아웃(time-out)이 발생한다. 타임아웃이 발생하면 바로 다음 포트로 이동하여 같은 절차에 따라 감지하게 된다. 만약 감지가 되면 이를 모듈보드로 인식하고, 모듈보드로부터 정보를 기다린다. 한편, 모듈보드에서는 현재 입력된 신호가 체크 신호인지 데이터 전송 신호인지를 인터럽트를 통해 인식하고 이에 대한 응답을 즉시 내보낸다. 모든 것에 대한 정리가 끝나면 그 모듈보드는 메인보드로부터 디스플레이 데이터 및 처리 루틴을 전송 받아 실행하게 된다. 이러한 기본 방식에 따라 시스템을 확장하게 되면 상당한 확장성을 보유하게 된다. 다음 작업으로 디스플레이 보드를 구성한다. 이때 신호를 체크하여 정보를 요구하는 신호일 경우 자기정보를 구성하게 되고 그 정보를 자신을 호출한 시스템에 전송한다. 메인보드에서 이에 따른 인터럽트를 발생시키고 발생된 인터럽트를 통해 모듈보드에 명령을 주는 서비스를 하게 된다. 본 시스템에서는 7-segment의 동작으로 검사하였다. 이러한 부분이 최근까지의 제어보드에 적용되어지지 않고 있었다. 뿐만 아니라 교육용 보드에서 응용되어지지 않았던 것은

제작자들이 무조건 주어진 명세서에만 의존하는 경향이 컸기 때문인 것으로 파악된다.

타 시스템 연결 정보		기존 데이터 유무	2개 이상의 장치	기능 인덱스 데이터			
0	0	0	1	0	0	1	1

그림 4. 전송데이터의 표현
Fig. 4. Representation of transmit data

본 시스템의 소프트웨어로 구현하는 부분에서 가장 중요한 것이 규약의 구현이다. 규약은 이미 설계부분의 그림 4에서 설명하였듯이 자기정보를 구성한다. 자기정보는 총 8비트의 데이터로 표현하기로 한다. 이것은 커넥터를 14핀으로 사용하지만, 8051 계열의 프로세서가 8비트 단위의 처리를 기본으로 하기 때문이다. 이에 대한 구현은 다음과 같다. 첫 번째 임의의 기능을 가진 객체형 보드의 경우를 4비트 내의 경우의 수로 제한하여 이들에 대한 인덱스를 표현하는 방법을 사용하였다. 본 시스템에서는 디스플레이 기능만을 하도록 구현하였다. 따라서 FND의 값을 세트시키는 것이다. 두 번째는 객체형 시스템의 주변장치 개수를 파악하는 것으로 기능 정보에 표현한 장치가 몇 2개 이상인지 아닌지를 0과 1로 표현하도록 하였다. 본 시스템에서는 1로 표현된다. 세 번째는 객체형 시스템에 디폴트 값으로 세팅된 작업이 있거나 이전 시스템과의 연결에서 저장된 데이터가 있는가에 대한 정보를 표현한다. 이 부분은 기존 작업이 있는 경우 이를 무시하고 클리어 후 새로운 작업 데이터를 전송 받을 수 있도록 표현하였다. 네 번째는 객체형 시스템에 다른 객체형 시스템이 연결되어 있을 수 있으므로 이에 대한 연결 개수를 2 비트로 표현하여 호출 시스템에 전달하도록 하는 부분이다. 이러한 과정에서 본 시스템은 0x13으로 표현되어졌다.

이제 모든 수행은 인터럽트가 서비스하도록 하였다. 인터럽트가 걸리면 해당 플래그가 세트되고 인터럽트 루틴이 수행된다. 8051 마이크로프로세서에는 인터럽트를 발생하는 하드웨어 구조가 5종류(8052는 타이머 2 가 추가되어 6종류)가 있으며, 각각의 인터럽트 요구에 대해서, 서비스 프로그램의 시작번지가 정리되어 있다. 인터럽트의 구동을 위해서는 IE레지스터와 IP레지스터를 참조하여야만 한다. 먼저 IE레지스터는 6개의 플래그로 구성되어 있으며, 비트 단위로 액세스가 가능하다. 그 중 5개의 플래그는 5개의 인터럽트 요청에 대해서, 인터럽트 인에이블 또는 디세이블을 제어한다. 남은 한 개의 플래그는 개개의 인터럽트 인에이블 / 디세이블을 제어하는 것이 아니고, CPU가 인터럽트 자체를 받을 것인가, 받지 않을 것인가를 제어하는데 사용한다. IP레지스터는 인터럽트 우선순위 제어 레지스터의 각 비트를 셋 또는 클리어 시키면, 그에 대응되는 인터럽트 요구에 대해서, 높은 레벨 혹은 낮은 레벨의 둘 중의 하나의 인터럽트 우선순위가 결정된다. 즉, 현재 낮은 레벨로 설정이 되어 있는 인터럽트 서비스 중에, 높은 레벨 우선순위 인터럽트 요구가 발생하면, CPU는 낮은 레벨의 인터럽트 서비스를 중단하고, 높은 레벨의 인터럽트 서비스를 시작한다. 실제 인터럽트에 대한 처리는 프로그램에서 하게 된다. 외부인터럽트 int0, int1은 INT0와 INT1 핀으로 입력한다. 사용자는 입력 신호의 검출 방식을 높이 검출 또는 모서리 검출 중에서 선택할 수 있다. 이 선택은 타이머 제어 TCON 레지스터의 인터럽트 타입 IT0와 IT1 비트로 설정한다. main() 함수에서는 EA를 세트시키고 Init_ExInt()와 Init_Timer()를 호출하여 인터럽트를 초기화시킨다. 이후에 폴링 방법을 이용하여 계속적인 연결 체크 동작을 수행한다. 이 작업은 계속되어지며, 이때 타이머 인터럽트가 같이 호출되어 루프를 돌게 된다.

그림 5은 C-51 프로그램으로 타이머 인터럽트를 사용하기 위해 초기화하고 서비스하는 부분의 알고리즘을 보여주고 있다.[6-7]

```
void Init_Timer()
{
  TMOD = 0x02;
  TH0 = 0x50;
  TR0 = 1;
  ET0 = 1;
}
interrupt void T0_int(void)
{
  if( port_num <= total_port_num )
    Send_Data_to_Module_board( port_num ) ;
    port_num ++ ;
  else
    T0_int disable ;
    EX0_int disable ;
}
```

그림 5. 타이머 인터럽트 프로그램
Fig. 5. Timer Interrupt program.

타이머 인터럽트 부분에서는 각각의 포트에 모듈보드의 연결 유무를 판단하기 위한 데이터전송이 주목적이 된다. 타이머 인터럽트를 사용한 이유는 폴링방법을 이용하여 각각의 포트를 조사하기 때문에 폴링에 대한 시간지연이 필요하기 때문이다. 타이머 인터럽트가 발생하면 모든 포트의 검사 유무를 확인하고 검사하지 않은 포트가 있을 경우 포트 쪽으로 포트정보요청을 위한 데이터를 전송하고 다음 포트의 검사를 위해 포트 번지를 증가시키게 된다. 만일 모든 포트의 검사가 완료되었다면 타이머 인터럽트와 외부 인터럽트를 모두 종료시키게 된다. 타이머 인터럽트에서 모듈보드로 보낸 신호에 의해 모듈보드가 응답신호를 보내게 되면 메인보드는 외부인터럽트가 걸리면서 인터럽트 서비스 루틴이 호출된다.

그림 6은 외부 인터럽트의 수행과정을 나타내는 알고리즘을 나타낸 것이다.

그림 6에서 Receive_Board_information() 함수는 모듈보드에서 보내는 정보를 받아서 리턴 해 주는 함수이다. 모듈보드에는 자기정보에 대한 데이터 값을 포트로 보내주게 되고 이것을 메인보드가 받아서 그 정보를 저장하게 되는 것이다. 메인보드는 저장된 정보를 가지고 이후 작업에서 모듈보드가 연결된 포트를 기억하여 모듈보드가 작업해야할 정보 데이타를 전송하여 모듈보드가 자신의 작업을 수행할 수 있도록 한다. 이러한 PNP와 인터럽트를 적절히 구현한 기능은 최근까지의 제어보드에 적용되어지지 않고 있었다.

```
type def board_feature[port][3] ;
void Init_ExInt(void)
{
  IT0 = 1;
  IE = 1;
}
interrupt void EX0_int(void)
{
  type def module_board =
Receive_Board_information( ) ;

board_feature[port_num][0]=module_board->port ;

board_feature[port_num][1]=module_board->kind
;

board_feature[port_num][2]=module_board->Data_
type ;
}
```

그림 6. 외부 인터럽트 프로그램
Fig. 6. External interrupt program

Ⅳ. 실험 및 분석

본 시스템은 모듈보드를 가장 많이 보유하고 있고 기존의 교육용 시스템 키트로 많이 사용되어지고 있는 Z80 계열의 시스템과 비교 및 분석하였다. 분석시 중요 포인트는 Z80을 중앙 처리 장치로 사용하고 있는 시스템의 구현 기간과 프로그램을 통한 디스플레이 기능에 대한 실행속도, 그리고 비용과 직결되는 구성장치의 수 및 보드의 크기가 고려되었으며, 마지막으로 기능적인 측면에서 그 복잡성을 분석하였다.

표 1. 기존 교육용 제어보드 Z80 SEC와의 비교
Table 1. Compare Z80 SEC with SEM system

세부내용	Z80 SEC	SEM 시스템
구 현	6MM	1MM
실행속도	0.6μ/sec	0.2μ/sec
장 치	3개의 I/O ADC, ROM 등 다수	4개의 I/O 필요에 따른 추가
확 장 성	Fix	Unfix
Hardware	램, ADC, 롬라이터 등	FND, MPU
Software	모니터링, 통신	모니터링, 데이터처리, 정보관리, 정보생성, 서비스

표 1.은 기존의 교육용 제어보드와 본 제어보드를 비교한 것이다. 제작 속도 관점에서 보면 구현을 위해 투자되어야 할 시간이 6MM 정도가 걸릴 것으로 추측되어 진다. 이것은 실제 현장에서 근무하고 있는 사람들과 이들이 비슷한 규모의 제어보드를 완성하는데 걸리는 시간을 기본 바탕으로 산출한 것이다. 이에 비해 SEM의 경우는 1명이 약 한달만에 완성하였고, 실제 특정 수준의 사람이 아니더라도 1달 정도 교육을 받으면서 구현하여도 충분히 가능하다. 또한 디스플레이용 프로그램을 작성하고 실행시키는데 있어서 큰 차이를 보이고 있는 것을 알 수 있다. 발광 다이오드에 신호를 주어서 점멸시키는 프로그램을 처리하는데 걸리는 시간이 두 시스템이 11.0592MHz에서 위와 같은 차이를 보이고 있다. 프로그램의 전송에 앞서 타이

머를 작동시키는 루틴을 주었고, 이 루틴이 끝남과 동시에 스톱워치와 같이 시간을 저장하고 출력할 수 있도록 처리하여 산출한 결과이다. 장치의 수를 비교하는 것은 약간의 모호성을 가진다. 왜냐하면 Z-80 SEC의 경우 지정된 장치들을 인식하고 통신할 수 있는 기본적인 하드웨어 장치들이 필요하기 때문이 설계 장치들이 많은 것이 당연하다. 이와는 반대로 SEM은 하나의 모듈이라는 전제로 필요한 부분은 추가하여 제작하고 지금 현재 실험할 수 있는 기능만을 구현하게 되므로 장치들의 수가 적은 것이다. 다음의 비교사항에서도 비슷한 성격을 가진다. 무엇보다도 SEM이 강조하고 있는 것은 확장성이다. 개발된 모듈의 경우 메인보드에서 약간의 복잡성을 가진다는 문제점은 있지만 이것을 다른 시스템에도 적용할 수 있도록 재사용 성을 가진다는 것이다.

V. 결 론

본 논문에서 제안한 방법은 시스템 개발 측면에서 기본보드의 구성을 최소화하였기 때문에 단순할 뿐만 아니라 개발이 쉬워지는 이점을 제공했다. 뿐만 아니라 보드의 기능 확장을 위해 모듈단위로 보드를 구현함에 따라 기능에 따른 보드의 확장성을 제공할 수 있었다. 이것은 하나의 메인보드에는 기본적인 기능과 함께 확장 포트만을 적용하고 그 외의 필요한 장치들을 적절한 요소에 따라 개발함으로써 유연성 및 높은 확장성을 가질 수 있었다. 시스템 적용 및 활용 면에서는 다양한 어플리케이션의 구축이 가능하고, 높은 이식성을 요구하는 시스템에 적합하다는 것이다. 이식성은 재 사용성을 제공하게 되고 나아가 자원의 소비절감에도 도움이 될 수 있다. 본 논문에서 제안한 객체형 시스템 아키텍처 방법은 좀 더 심화된 연구를 걸쳐서 실제 구현이 가능하도록 만들어진다는 전제에 있다.

본 논문에서 제안한 시스템에서는 외부 모듈보드의 인식과정에서 약간의 시간이 지연되는 결과가 발생하는 단점이 나타났다. 이러한 시간지연의 문제를 해결하기 위해 본 제어 보드에 시스템 내부에 운영체제(RTOS)의 개념을 도입하여 모듈의 인식과정과 통신을 위한 태스크를 이용한다면 그 확장성 및 유연성이 더 확대되면 시간 지연에 대한 문제도 해결될 수 있을 것으로 생각된다.

참 고 문 헌

[1] D. Gajski, "*A Methodology for Easy&Efficient reuse of IP and hardware/software co-design*", Oct 1996.

[2] 김남훈, "*New partitioning techniques in hardware-software codesign*", 전자공학회 논문지, 제35권 C편, 5호, pp.1-10, May. 1998.

[3] Daniel D. Gajski, Gaurav, Aggarwal, En-Shou Chang, Tadatoshi Ishii, "*Methodology for Design of Embedded systems*", Mar. 1998.

[4] 김병철 외 5명, "*TCP/IP 프로토콜*", pp.26-30, 미래컴, Aug. 2000

[5] http://home.opentown.net/~manseok/micro

[6] 인텔 제어 연구회, "i8051 C 실습", 전체, Ohm 사

[7] 정용원, "*8051 기초 + α*",전체, 성안당

[8] Lee, Moon-Key, "A VLSI *implementation of 32-bit RISC embedded controller*", 전자공학회논문지, 제31권 A편, 10호, pp.141-151, Oct. 1994.

[9] 신현철, "*하드웨어-소프트웨어 통합 설계 기술*", 전자공학회지, 제24권 12호, pp.69-78, Dec. 1997.

저 자 소 개

印 致 虎 (正會員)

1985년 한양대학교 전자공학과 공학사 1987년 한양대학교 대학원 공학석사(VLSI CAD 전공) 1996년 한양대학교 대학원 공학박사(VLSI CAD 전공) 1992년 ~ 현재 세명대학교 컴퓨터과학과 부교수 e-mail : ich410@semyung.ac.kr

<관심분야> VLSI CAD, ASIC 설계, CAD 알고리듬, RTOS 및 내장형 시스템