ROM BIOS를 이용한 컴퓨터 하드웨어 장애인식 모듈 설계

남의석[1*]
[1]극동대학교 유비쿼터스IT학과

Design of Computer Hardware Fault Detector using ROM BIOS

Eui-Seok Nahm[1*]
[1]Department of Ubiquitious IT, Far East University

요 약 PC에서 모니터나 본체에서 전원인가 후 아무런 외부적인 반응이 없는 경우, 이것이 하드웨어 장애인지 아니면 소프트웨어 장애인지 일반 사용자의 경우 확인이 어려운 상황이다. 본 연구는 이러한 경우 컴퓨터 하드웨어 장애 인지 소프트웨어 장애인지를 구별하고 하드웨어 장애인 경우 어떤 하드웨어가 장애인지를 알려주는 모듈을 개발하였다.
즉, 본 연구는 일반 PC의 OS 부팅 전에 하드웨어 장애로 인한 부팅 실패시 하드웨어 장애를 인식하는 모듈을 개발한 것으로 컴퓨터 PCI slot에 장착되며 컴퓨터 전원 인가시 CPU는 메인보드 롬바이어스 내용을 순차적으로 처리하는데 특정 하드웨어 장애시 순차적 진행을 중단한다. 본 연구의 개발 모듈은 이러한 경우 롬바이어스의 중단 번지를 인식하여 장애 하드웨어를 사용자에게 알려주는 역할을 한다. 개발된 하드웨어 장애인식 모듈은 모의 평가 시험을 통해서 정확하게 인식할 수 있었다.

• **주제어** : 롬바이오스, 하드웨어결함, 부팅, 컴퓨터, 인식

Abstract Currently almost people use a personal computer for various purpose. But some people are not familiar to computer system. If they see only black screen on the monitor when they turn on the computer power, they can not recognize whether it is hardware or software faults. So, in this paper is aimed to develop the module of computer hardware fault detecter using ROM BIOS before OS booting. This module use PCI interface with mother board of computer. Before os booting, it can get the ROM BIOS memory by interrupt and show what hardware is fault according to the predefined memory content of BIOS.

• **Key Words** : ROM BIOS, Hardware Fault, Booting, Computer, Detector

1. 서론

PC에서 모니터나 본체에서 전원인가 후 아무런 외부적인 반응이 없는 경우, 이것이 하드웨어 장애인지 아니면 소프트웨어 장애인지 일반 사용자의 경우 확인이 어려운 상황이다. 보통의 경우 AS센터로 하드웨어를 전체를 보내거나 아니면 수리 전문가가 방문하여 수리하는 것이 일반적이다. 이 경우 시간과 비용이 들어간다. 본 연구는 이러한 경우 PC 장애의 원인을 일반 사용자가 쉽게

*교신저자 : 남의석(nahmes@kdu.ac.kr)
접수일 2013년 8월 16일 수정일 2013년 9월 3일 게재확정일 2013년 9월 4일

판단하여 대응할 수 있도록 하는 하드웨어 장애 인식 카드를 개발하는 것이다. 즉, 모니터 장애, 그래픽 카드 장애, 또는 마더보드 장애, 랜카드 오류, 사운드 카드 오류 등 이러한 정보를 부팅시 알려주도록 하는 기능을 하는 카드를 개발하는 것이다.[1][2]

하드웨어 장애인 경우 전문적인 지식을 소유한 일부의 전문가 집단이나 전문 수리점에서 확인할 수 있었던 수준의 장애 정보를 제공함으로서 사용자들이 직접 자신의 PC를 진단함으로서 보다 신속하고 정확한 사후 조치를 할 수 있게 되어 그로 인해 발생하는 비용 및 시간 등을 절약할 수 있고, 전문적인 조립 PC판매처나 유지 보수 서비스 제공자의 경우에는 고객에게 보다 신뢰도 있는 서비스를 제공함으로서 방문고객의 만족도를 향상 시킬 수 있고 이로 인한 재구매 고객 확보 및 판매 증가 및 수리시간의 단축으로 인한 효율적인 인력 운용 등 긍정적인 파급 효과를 기대 할 수 있다.[3][4]

현재의 각 가정 및 기업의 PC 보급의 증가세를 생각해 볼 때 그에 비례해야하는 전문 수리 또는 유지보수 인력은 턱없이 부족하다. 또한 통상 유지 보수 서비스업체와 유지 보수 계약이 체결되어 있는 기업의 업무용 PC의 경우에도 대당 책정된 유지 보수 비용은 유지 보수 업체들간의 과당 경쟁으로 인한 저가공세로 인해 사용자들의 기대수준을 충족시킬 수 있는 서비스를 기대하기 어렵다.

본 논문에서는 일반 PC의 OS 부팅 전에 하드웨어 장애로 인한 부팅 실패시, 장애 하드웨어에 대한 정보를 ROM BIOS 통해 정보를 읽어 사용자에게 제공하는 PCI 기반의 하드웨어 장애 감지 카드(모듈) 개발하고자 한다. 개발된 하드웨어 장애인식 모듈은 모의 평가 시험을 통해서 정확하게 인식할 수 있었다.

2. 기존 PC 하드웨어 장애인식 및 처리 방법

실제로 눈에 보이는 증상이 없는 한 PC에 어떠한 문제가 있는 지를 판단한 다는 것은 불가능한 일이다. 특히 모니터나 본체에서 전원인가 후 아무런 외부적인 반응이 없는 경우는 유지 보수 서비스 업체에게 PC본체를 통째로 맡기는 수밖에 없다. PC 방문수리는 회당 6만원 이상의 비용이 들어 PC사용이 불가능해질 때까지 방치하는 경우가 많았다. 현재의 일상적인 사무 환경에서 사용 중인 PC의 장애란 곧 그 PC 사용자의 업무 마비일 정도로 의존도가 높고 또한 그에 따른 관련 부서의 업무지장을 모두 감안 한다면 결코 PC의 소리 없는 장애란 작은 문제가 아닌 것이다.

물론 현재는 PC 자체의 가격이 저렴해져서 수리보다는 교체쪽으로 문제를 해결하는 경우도 많이 있기는 하지만 PC 한 대를 자신이 사용하던 형태로 설치 및 복구한다는 것도 상당한 시간과 노력이 필요하고 남이 대신 해줄 수 없는 부분도 있어서 단순히 구매로 대체 하는 것은 어디까지나 한계가 있는 것이다. 유지 보수 서비스 제공처를 찾는 것도, 찾아서 출장기사를 부르는 것도 모두 시간과 비용이 상당히 소모되는 일일뿐더러 원하는 시간에 원하는 서비스를 받는 것도 현실적으로 불가능한 일이다.

또한, 출장기사나 유지보수 서비스 제공처에서도 일정 수준이상의 장애에 대한 정확한 진단을 하는 대에는 특별히 증상을 보이지 않는 위와 같은 경우에는 원인을 파악하는 데에는 상당한 시간을 소요하게 될 수밖에 없다. 이러한 PC의 장애가 확인 결과 단순한 접촉의 불량이거나 또는 간단한 이물 질제거로 문제가 해결된 수도 있다면 장애의 원인을 사용자가 쉽게 확인하는 것이 상당히 의미가 있는 것이다. 마치 소화기의 용도와 마찬가지로 언제 사용하게 될지 모르지만 그 한번을 위해 보험을 드는 것이다. PC 사용자들의 생산성이 제고되면 그만큼 회사나 국가적으로도 이익이 될 수 밖에 없다.

특히, 선진국의 경우는 높은 인건비로 인해 출장 서비스는 거의 불가능하고 사용자 자체의 인건비도 비싼 편이어서 업체에 납품되는 PC에 기본사양으로 채택될 가능성도 국내보다 높다.

3. 본 논문의 PC 하드웨어 장애인식 및 처리 방법

본 연구는 일반 PC의 OS 부팅 전에 하드웨어 장애로 인한 부팅 실패시, 장애 하드웨어에 대한 정보를 사용자에게 제공하는 PCI 기반의 하드웨어 장애 감지 카드(모듈) 개발을 목표로 한다. 개발 카드는 컴퓨터 PCI slot에 장착되며, 컴퓨터 전원 인가시 CPU는 메인보드 롬바이어스 내용을 순차적으로 처리하는데 특정 하드웨어 장애시 순차적 진행을 중단한다. 본 개발카드는 이러한 경우

롬바이어스의 중단 번지를 인식하여 장애 하드웨어를 사용자에게 알려주는 역할을 한다.

본 논문의 개발카드 사용시 장애 인식 하드웨어의 범위 및 종류는 다음과 같다.

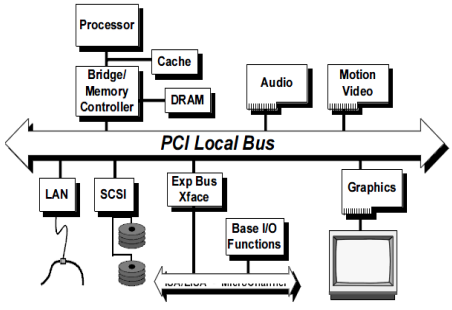
☑ CPU 교환, 메모리, 메인보드 교환
☑ 메모리 교환, 메모리 슬롯 청소, 메인보드 교환
☑ 비디오카드, 메인보드 교환
☑ 밧데리 점검, 메인보드 교환
☑ CPU, 메인보드 교환
☑ 메인보드 교환
☑ 메모리 교환, 메모리 슬롯 청소, BIOS, CPU
☑ IO 칩, 메인보드 교환
☑ 키보드, 메인보드 교환
☑ 마우스, IO 칩 교환
☑ 사운드
☑ BIOS 교환
☑ BIOS 교환, 메인보드 교환
☑ 밧데리, CMOS 회로, 파워리플증가
☑ 메인보드 교환, 파워
☑ 비디오카드, 메인보드, VGA 회로
☑ 비디오카드
☑ 마우스, 메인보드 교환
☑ 메모리, 메인보드 교환
☑ 메인보드 교환, CPU 검사
☑ 비디오카드, 메인보드 교환
☑ CPU 교환
☑ CMOS 설정
☑ Setup 재설정, BIOS 교환
☑ 플로피, 메인보드 교환
☑ HDD, 메인보드 교환
☑ HDD, BIOS 교환, CMOS 재설정

본 논문의 개발기술은 PC에 장착되는 하드웨어 모듈(PCI Type)과 하드웨어별 롬바이어스 번지 이식 장애 내용 표시 소프트웨어로 구성되어 있다. PC에 장착되는 하드웨어 모듈(PCI Type)를 개발하고 이 하드웨어에 최적화된 응용프로그램 및 관리 소프트웨어 개발하는 단계로 진행된다. 가장 중요한 일은 각 개발품에서 소프트웨어의 연산량 등의 기술개발에 시스템의 성능과 상세 필요기능을 설계하는 일이다.[6][7]

바이어스란 'Basic Input/Output System'의 약자로, 그대로 해석하자면 기본적인 입력과 출력 담당하는 시스템'을 의미한다. 바이어스는 입출력에 관계된 대부분의 하드웨어에 존재하며, 특별한 경우 외부에서 이를 지원하는 형태로 되어 있기도 하다. 요컨대, 하드웨어는 사용자의 요구에 의해 데이터를 입출력하므로 이런 동작을 수행하기 위해 하드웨어를 제어하는 가장 기본적인 코드를 필요로 하는데, 이런 일을 담당하는 것이 바로 바이어스인 셈이다. 한편으로 '펌웨어(Firm Ware)'를 의미하기도 한다. 펌웨어는 하드웨어와 소프트웨어의 중간 형태로 존재하는 프로그램이다. 하드웨어 설계시 그 시스템의 가장 기본적인 수행을 위한 코드는 특별한 형태의 프로그램으로 만들어져 하드웨어 내부에 내장되기 때문에 제작 당시 넣은 형태에서 변경이 어렵다. 또한 소프트웨어가 하드웨어를 제어하려면 펌웨어를 거쳐야 하기 때문에 프로그램이지만 좀 다른 모습을 하게 된다. 따라서 바이어스라는 말은 하드웨어와 구별하여 펌웨어라는 말과 혼용해서 사용하기도 한다. 가장 하위층이 하드웨어, 가장 상위층이 소프트웨어이며, 그 중간을 연결하는 매개체가 펌웨어 또는 바이어스이다. 펌웨어는 마더보드, 비디오카드, 기타 다른 여러 장치 등과 같은 하드웨어에 있는 메모리 중 ROM에 저장되어 있으며, PC가 켜지면 필요한 여러 루틴들과 자주 사용되는 기본 동작에 쓰이는 프로그램과 데이터를 제공하게 된다. 하지만 펌웨어를 특정한 ROM의 형태가 아니라 칩 설계시 내부에 메모리를 설정하고 그곳에 올라가도록 하는 경우도 있으며, 플래시 ROM을 사용하여 재기록이 가능 하도록 함으로써 버그 패치와 성능 향상을 위한 업그레이드가 가능한 제품들도 있다. 컴퓨터를 구성하는 가장 중요한 부속물은 마더보드 혹은 메인보드라고 불리는 기판이다. 마더보드에는 CPU와 RAM이 설치되어 있으며, 다수의 애드온 카드들이 연결될 수 있다. 또한 외부 장치와의 연결을 위해 커넥터가 제공된다. 마더보드는 이 모든 것을 컨트롤하는 역할을 담당하므로 이것을 총괄할 매개체가 필요하며, 그 역할을 담당하는 것이 바이어스이다. 바이어스는 시스템을 사용할 수 있도록 하기 위하여 다음 단계로 넘어가게 되며, 마더보드에 장착된 각 칩셋과 연결되어 있는 여러 장치들에 대해 각각 초기화를 시작한다. 초기화 과정의 일환으로써 ROM 바이어스 내부에 들어 있는 인터럽트 핸들러를 읽어 들이게 되고, 인터럽트 벡터 테이블을 구성한다. 또한 현재 시스템에 장착되어 있는 장비들의 상태를 알아내 메모리의 하위 번지에 기록하고, 확장

바이어스의 유무를 점검하여 확장 바이어스가 있는 장치의 메모리 하위 번지에 확장 바이어스를 설치한다. 요즘 자주 사용되는 SCSI 컨트롤러를 예로 들면, SCSI카드에 있는 ROM 바이어스가 컴퓨터의 메모리에 설치되었다는 메시지를 볼 수 있을 것이다. 물론 이 메시지는 SCSI카드에서 리포트하며, 경우에 따라서는 리포트하지 않는 SCSI카드도 있다. 이렇게 설치된 SCSI 바이어스는 확장 바이어스의 대열에 들어가 시스템의 ROM 바이어스 계열에 합류하게 된다. 물론 이 과정은 시스템에 전원이 들어갈 때마다 매번 실행된다. 시스템 초기화 과정까지 아무 문제가 발생하지 않으면 바이어스는 디스크 부팅을 준비하며, 이 작업을 위해 부팅할 디스크로부터 부트스트랩 로더를 읽어 들인다. 부트스트랩 로더는 시스템 부팅에 필요한 아주 작은 프로그램으로서, 이것을 읽는 데 성공하면 디스크 부트가 시작된다. 디스크 부트의 시작과 더불어 디스크에 설치된 운영체제가 컴퓨터의 메모리에 올라오며, 운영체제는 시스템을 점검하고 일련의 초기화 과정을 수행한 다음 사용자의 명령을 기다린다. 이후로 사용자는 프로그램을 사용한다든가 시스템을 직접 제어하는 등의 작업이 가능하다. 물론 이 상태에서도 바이어스는 계속 사용되며, 컴퓨터 작동에 필요한 여러 루틴들이 함수 형태로 제공된다. 사용자는 이것을 호출에 의해서 사용할 수 있으며, 운영체제에 의해서도 사용할 수 있다.[1][5]

본 논문의 개발은 PCI 방식으로 마더보드와 인테페이스 된다. 아래 그림 1은 PCI 인터페이스 개념도이다.

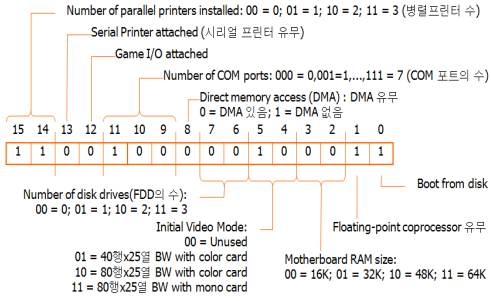


[Fig. 1] Block Diagram of PCI System

본 연구는 PCI 인터페이스를 통하여 이러한 바이어스를 검사하는 일련 과정들을 개발 보드가 감지하여 장애를 표시하여 사용자가 쉽게 장애 원인을 알 수 있도록 하는 것이다. 그림 2는 ROM BIOS 접근 프로그램을 보여준다.

```
01  #include <stdio.h>
02  #include <dos.h>
03  int main()
04  {
05    struct BIOS_EQUIP {
06      unsigned bootdisk  : 1;    /* 디스크 부팅 유무 */
07      unsigned copro     : 1;    /* 코프로세스 탑재 유무 */
08      unsigned ram       : 2;    /* 마드보더의 램 크기 */
09      unsigned video     : 2;    /* 비디오 모드 */
10      unsigned fdd       : 2;    /* FDD의 수 */
11      unsigned dma       : 1;    /* DMA 장착 유무 */
12      unsigned comport   : 3;    /* COM Port의 수 */
13      unsigned gameio    : 1;    /* GAME I/O 유무 */
14      unsigned s_prt     : 1;    /* 시리얼 프린터 유무 */
15      unsigned p_prt     : 2;    /* 병렬 프린터의 수 */
16    } *equip;
17    unsigned equipment;
18    equipment = peek(0, 0x410); /* 0000에서 0410만큼
      떨어진 곳의 2바이트 읽기 */
19    equip = (struct BIOS_EQUIP *)&equipment;
20    printf("<< ROM BIOS DATA CHECKING >> \n");
21    printf("Boot from disk = %u \n", equip->bootdisk);
22    printf("Floating-point Coprocessor = %u \n",
equip->copro);
23    printf("Motherboard RAM size(kb)     = %u \n",
(equip->ram+1)*16);
24    printf("Initial VIDEO Mode = %u \n", equip->video);
25    printf("Number of disk drives(FDD) = %u \n",
equip->fdd);
26    printf("Direct Memory Access(DMA) = %u \n",
equip->dma);
27    printf("Number of COM ports = %u \n",
equip->comport);
28    printf("Game I/O attached = %u \n", equip->gameio);
29    printf("Serial Printer attached = %u \n", equip->s_prt);
30    printf("Number of Parallel printers installed = %u \n",
equip->p_prt);
31    return 0;
32  }
```

[Fig. 2] Program of ROM BIOS Addressing



[Fig. 3] 2 Byte Device Information of ROM BIOS (0410 ~ 0411)

여기서 ROM BIOS의 장치 정보를 읽어 들여서 정보를 표시하는 것으로 그림 3은 ROM BIOS 2Byte 장치 정보를 보여준다. 본 연구는 이러한 ROM BIOS의 정보를 읽어 하드웨어 장애를 표시해 주는 것이다.

여기서 또한 이러한 ROM BIOS의 정보를 읽어 들이는 방법은 인터럽트 서비스 루틴 가로어야 한다. 이러한 방식은 그림 3과 같은 방식으로 한다.

① AH 레지스터에 인터럽트 서비스 번호를 할당한다.
② 기타 레지스터(BX, DX)에 필요한 데이터를 할당한다.
③ 인터럽트 수행 명령 INT를 사용하여 인터럽트 서비스 루틴을 호출한다.
④ 각 레지스터에 반환된 반환 값을 얻어 수행 결과를 확인한다.

```
MOV AH, 0E    /* 인터럽트 10h의 기능 중 문자출력 서비스 번호 0E를 넣는다. */
MOV AL, 65    /* 출력할 문자의 데이터(ASCII Code)를 AL 레지스터에 할당한다. */
MOV BH, 0     /* 출력할 비디오 화면의 페이지 번호를 할당한다. (0번: 현재페이지) */
MOV BL, 1     /* 출력할 문자의 색깔을 할당한다. */
INT 10        /* INT 명령으로 인터럽트 서비스 루틴 10h번을 실행한다. */
```

[Fig. 4] Interrupt of ROM BIOS Service Routine

이러한 ROM BIOS의 서비스 루틴 가로채기 로직은 그림 5와 같다.

```
01  #include <stdio.h>
02  #include <dos.h>
03  void my_putch(char ch)
04  {
05    union REGS r;
06
07    r.h.ah = 0x0E;
      /* 문자 출력을 위한 서비스 번호를 AH 에 대입 */
08    r.h.al = ch
      /* 출력할 문자를 AL 레지스터에 할당 */
09    r.h.bh = 0;    /* 현재 보여지는 화면에 출력 */
10    r.h.bl = 1;    /* 출력할 문자의 색상 */
11    int86(0x10, &r, &r);  /* 인터럽트 서비스 루틴 호출 */
12  }
13  int main()
14  {
15    char *t = "What a wonderful world!";
16    while (*t != 0) my_putch(*t++);
17    return 0;
18  }
```

[Fig. 5] Interrupt Logic of ROM BIOS Service Routine for Intercept

[Table 1] Information of ROM BIOS

인터럽트	어드레스	종류	설 명
00h	0000:0000h	Processor	Divide by zero
01h	0000:0004h	Processor	Single step
02h	0000:0008h	Processor	Non maskable interrupt (NMI)
03h	0000:000Ch	Processor	Breakpoint
04h	0000:0010h	Processor	Arithmetic overflow
05h	0000:0014h	Software	Print screen
06h	0000:0018h	Processor	Invalid op code
07h	0000:001Ch	Processor	Coprocessor not available
08h	0000:0020h	Hardware	System timer service routine
09h	0000:0024h	Hardware	Keyboard device service routine
0Ah	0000:0028h	Hardware	Cascade from 2nd programmable interrupt controller
0Bh	0000:002Ch	Hardware	Serial port service - COM post 2
0Ch	0000:0030h	Hardware	Serial port service - COM port 1
0Dh	0000:0034h	Hardware	Parallel printer service - LPT 2
0Eh	0000:0038h	Hardware	Floppy disk service
0Fh	0000:003Ch	Hardware	Parallel printer service - LPT 1
10h	0000:0040h	Software	Video service routine
11h	0000:0044h	Software	Equipment list service routine
12h	0000:0048h	Software	Memory size service routine
13h	0000:004Ch	Software	Hard disk drive service
14h	0000:0050h	Software	Serial communications service routines
15h	0000:0054h	Software	System services support routines
16h	0000:0058h	Software	Keyboard support service routines
17h	0000:005Ch	Software	Parallel printer support services
18h	0000:0060h	Software	Load and run ROM BASIC
19h	0000:0064h	Software	DOS loading routine
1Ah	0000:0068h	Software	Real time clock service routines
1Bh	0000:006Ch	Software	CRTL - BREAK service routines
1Ch	0000:0070h	Software	User timer service routine
1Dh	00000074h	Software	Video control parameter table
1Eh	0000:0078h	Software	Floppy disk parameter routine
1Fh	0000:007Ch	Software	Video graphics character routine
20h-3Fh	0000:0080f - 0000:00FCh	SOftware	DOS interrupt points
40h	0000:0100h	Software	Floppy disk revector routine
41h	0000:0104h	Software	hard disk drive C: parameter table
42h	0000:0108h	Software	EGA default video driver

43h	0000:010Ch	Software	Video graphics characters
44h	0000:0110h	Software	Novel Netware API
46h	0000:0118h	Software	Hard disk drive D: parameter table
4Ah	0000:0128h	Software	User alarm
64h		Software	Novel Netware IPX
67h		Software	EMS support routines
70h	0000:01c0h	Hardware	Real time clock
71h	0000:01C4h	Hardware	Redirect interrupt cascade
75h	0000:01D4h	Hardware	Math coprocessor exception
76h	0000:01D8h	Hardware	Hard disk support
77h	0000:01DCh	Hardware	Suspend request
7Ah		Software	Novell Netware API

4. 평가 및 분석

본 개발 모듈의 기능을 점검하기 위하여 각 하드웨어 모듈별로 보드를 크랙하거나 또는 특정칩을 강제로 장애를 발생하여 평가하였다. 아래 표 2는 하드웨어별 장애발생 방법과 장애 인식 평가 결과를 보여준다.

[Table 2] Result of Fault Detect

하드웨어	장애 발생 방법	장애 인식 결과
CPU	크랙 발생	정상 인식
메모리	보드 훼손	정상 인식
메인보드	장애 보드 설치	정상 인식
비디오 카드	보드 연결부 훼손	정상 인식
메모리 슬롯	크랙 발생	정상 인식
키보드	연결선 절단	정상 인식
마우스	연결선 절단	정상 인식
I/O Chip	장애 Chip 사용	정상 인식
사운드 카드	장애 Chip 사용	정상 인식
HDD	장애 HDD 사용	정상 인식

5. 결론

본 연구는 일반 PC의 OS 부팅 전에 하드웨어 장애로 인한 부팅 실패시 하드웨어 장애를 인식하는 모듈을 개발한 것으로 컴퓨터 PCI slot에 장착되며 컴퓨터 전원 인가시 CPU는 메인보드 롬바이어스 내용을 순차적으로 처리하는데 특정 하드웨어 장애시 순차적 진행을 중단한다. 본 연구의 개발 모듈은 이러한 경우 롬바이어스의 중단 번지를 인식하여 장애 하드웨어를 사용자에게 알려주는 역할을 한다.

개발된 모듈을 이용하여 실제 컴퓨터의 메모리, CPU, 비디오 카드, 메인보드, HDD, 사운드카드, IO 장치 등을 강제로 장애로 발생하여 성능을 검증한 결과 모두 장애를 정확히 인식한 것으로 모듈의 성능을 검증하였다.

그러나, ROM BIOS의 내용이 CPU의 발전에 의해 변경될 수 있기 때문에 이에 대한 개발 모듈의 S/W 자동 업데이트 기능이 향후의 연구에서 필요하다고 판단된다.

REFERENCES

[1] http://www.smartechconsulting.com
[2] http://www.basicinputoutputsystem.com/
[3] IBM ROM BIOS:PROG' QUICK REFE' SERIES
[4] http://www.answers.com/topic/rom-bios
[5] http://www.ehow.com/how_5183777_update-bios-rom.html
[6] http://www.brainbell. com/tutors /A+/ Hardware /_ROM_BIOS.htm
[7] Sb Rnice: PCI, Universal Serial Bus, Sata, RS-485, USB Hub, SCSI, Musical Instrument Digital Interface, Agp, I C, S Riov Komunik

저자소개

남 의 석(Eui-Seok Nahm) [정회원]

- 1993년 2월 : 연세대학교 전기공학과 (공학석사)
- 1998년 2월 : 연세대학교 전기공학과 (공학박사)
- 1998년 3월 ~ 2002년 5월 : LS산전 중앙연구소 선임연구원
- 2003년 3월 ~ 현재 : 극동대학교 유비쿼터스 IT학과 교수

<관심분야> : 지능형 제어, 홈네트워크