韓國 OA學會 論文誌
第7卷 第2號, 2002. 6.

2002-7-2-3-7

임베디드 STEP 컨버터의 개발에 관한 연구

최 준 기*

A Study on the Development of Embedded STEP Converter

Joon-Kee Choi*

요 약

최근 인터넷이 발달함에도 불구하고 신형 장비들은 이더넷 포트를 내장하고 있으나, 구형 장비들은 시리얼 포트가 내장되어 있는 경우가 많다. 따라서, 이를 유지 보수하기 위해서는 원격지에서 모뎀을 통하여 작업을 하거나 직접 기기 앞에서 작업을 해야 하는 번거로움과 낭비가 있다.

본 연구를 통하여 시리얼 장비중 특히, 교환기들을 이더넷상에서 통제하고 관리할 수 있도록, 임베디드 보드에 실시간 리눅스 운영체제 및 기타 응용 소프트웨어를 구축한 실시간 임베디드 시스템을 개발한다. 테스트 결과 원격지에서 시리얼 교환기에 접속할 수 있었으며, 상태 체크 및 제어가 가능함을 확인하여 기타 시리얼 장비들에도 응용 가능함을 확인하였다.

Abstract

Recently, new appliances have become built-in ethernet port but most of the old devices have serial port. So, it complicates and needs long time to maintain at a long distance. It can only controllable and fixable using modem.

In this paper, we developed an embedded STEP converter within Linux operating system and other application software so that serial devices(PBX) can control in the ethernet network.

After completion of development processes, test was conducted. In the remote places, it was connected the STEP converter and controled the serial PBX. Finally, we confirmed that it can apply to other equipments.

* 인덕대학 여성정보행정과 조교수

논문접수 : 2002. 3. 29
심사완료 : 2002. 6. 7

Ⅰ. 서론

최근들어 각 가정이나 사무실 등에 인터넷 전용선이 설치되어 편리하게 정보를 수집할 수 있고 업무도 볼 수 있다. 이러한 환경은 불과 몇 년전만해도 상상할 수 없었던 일이며 최근 몇 년 사이에 놀랄만한 변화를 가져오고 있다.

많은 변화 가운데 가장 큰 변화중의 한 가지는 대부분의 기기들이 이더넷 환경에서 동작하도록 개발되고 만들어진다는 것이다. 그 이유는 여러 가지 편리성의 추구 및 멀티미디어 기능을 최대한 이용하려하기 때문이다.

최근에는 이를 뛰어 넘어 무선 개념을 추구하는 경향이 많지만 유선의 장점을 활용하려는 시스템도 아직까지는 많다. 특히, 공장자동화 분야에서 제어 계측 장비 등을 활용하려는 노력들이 많은데, 한 가지 문제점은 이더넷 환경과 접목할 수 없는 고가의 장비들이 아직도 무수히 많다는 점이다. 장비를 대체하기에는 막대한 비용이 필요하고 그냥 사용하기에는 불편한 점이 많아 회사들로서는 어려운 처지에 놓여 있는 것이 사실이다. 만일, 중간에 이더넷 환경과 연결시켜줄 연결 장치가 있다면 여러 가지 활용도 측면에서 많은 이점이 있을 것이다. 임베디드 시스템(Embedded System)이 그러한 역할을 해 줄 수 있다.

기존에 만들어진 고가의 장비들은 시스템마다 환경이 다르지만 시리얼 라인을 통하여 자료를 송수신하고 있는 점은 비슷하다.

본 연구에서는 이러한 기존 장비들의 특성을 활용하여 시리얼 라인과 이더넷 라인을 연결시켜 자료의 송수신이 가능하게 하고 접근 및 제어가 용이한 임베디드 시스템을 개발하고자 한다. 오픈 운영체제인 실시간 리눅스 운영체제를 이용하기 때문에 개발 과정이 다소 복잡할 수 있지만, 개발 비용이 저렴하고 개발이 완료되면 기존 시리얼 장비들도 이더넷 상에서 자유롭게 제어 및 관리가 가능하리라고 본다.

Ⅱ. 임베디드 시스템

임베디드 시스템은 하드웨어와 소프트웨어가 결합된 형태로 응용 범위가 다양하다. 이 시스템은 미리 정해진 규칙대로 동작하도록 정의되어 있으며, 정해진 시간내에 처리가 완료되도록 구성된 시스템과 어느 정도의 정해진 시간 초과를 받아들이는 시스템으로 나누어 볼 수 있다. 최근들어 이 임베디드 시스템이 처리할 수 있는 일의 범위가 증대되고 있어 내장된 마이크로 프로세서의 성능이 증가하고 메모리의 양이 늘어나는 추세에 있다. 시스템에서 다양한 일을 처리하기 위해서는 운영체제를 필요로 한다. 임베디드 시스템에서의 운영체제는 필수적이며 그 기반위에 응용 프로그램들이 동작한다[3][4].

대부분의 임베디드 시스템은 시간 제약의 유무에 관계없이 실시간 성을 갖는다고 볼 수 있다. 시스템에서 이러한 실시간 성을 갖도록 운영체제가 여러 가지 관련 작업들을 해주게 된다. 이를 실시간 운영체제(Real Time Operating System)라고 한다. 실시간 운영체제는 OS의 관점에서 보면 선점형 멀티태스킹을 지원하고, 각 태스크들은 우선순위를 가지고 있어 높은 우선순위를 가진 태스크가 먼저 실행되도록 구성되어 있다.

현재는 많은 상용 운영체제들이 있지만 라이센스 비용이 필요없는 실시간 리눅스 운영체제가 임베디드 시스템 분야에서 자리를 잡아가고 있다. 오픈소스이며 POSIX를 지원하여 자유로운 응용이 가능하기 때문에 개발 비용 및 시간을 단축시킬 수 있다[5-7].

Ⅲ. 개발 개요

본 연구에서 개발하고자하는 임베디드 시스템은 시리얼 라인을 통해 관리 및 통제가 가능한 PBX를 이더넷 환경이 구축되어 있는 어느 곳에서나 인터넷에 접속하여

이를 제어할 수 있는 컨버팅 장비이다. 현재는 모뎀을 통해 PBX를 유지보수하고 제어하기 때문에 여러 가지로 불편하며, 유지보수 비용이 많이 들어가는 상황이다. 개발하고자 하는 임베디드 시스템을 이용하면 편리하게 교환기에 접속하여 원하는 작업을 수행하는 것이 가능하다. 또한, 신형으로 교체하는데 소요되는 막대한 비용을 절약할 수 있다.

전체적인 개념을 나타내는 구조도는 그림 1에서 보인다.

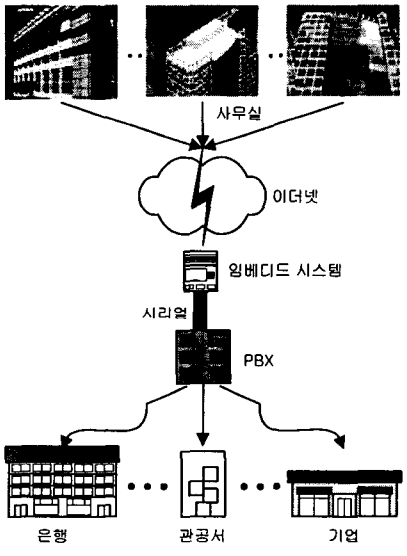

그림 1. 전체적인 구조도
Fig. 1 Overall Structure

그림 1에서 보는 것처럼 임베디드 시스템은 이더넷 라인과 PBX의 시리얼 라인을 연결해주는 컨버터 역할을 담당하며 원격지에서도 편리하게 시리얼 장비들을 관리할 수 있다. 다음 그림 2는 임베디드 시스템의 상세 구조도를 보인다.

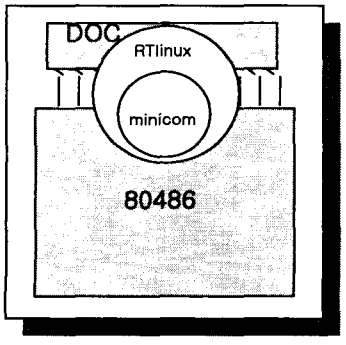

그림 2. 임베디드 시스템 상세 구조도
Fig. 2 Detailed Structure of Embedded System

X86 계열인 486 임베디드 개발자 보드에 일종의 플래시 메모리인 DiskOnChip(이하 DOC)을 탑재하고 여기에 각종 소프트웨어를 설치한다[2]. 임베디드 시스템의 특성상 적은 공간에 소프트웨어를 설치해야 하기 때문에, 최소의 RTlinux 커널과 루트 파일 시스템을 설치하고 통신 응용 프로그램인 미니콤 프로그램이 필요하다.

다음 장에서 좀 더 구체적으로 논하고자 한다.

Ⅳ. 실시간 STEP 컨버터 개발

실시간 STEP(Serial-To-Ethernet Protocol) 컨버터는 시리얼 교환기 등을 원격지에서도 실시간으로 제어하고 관리가 가능하도록 함으로써 유지보수 비용을 절약할 수 있고 시간 낭비도 줄일 수 있다. 개발 과정에 대한 전체적인 내용은 다음과 같다[8-15].

4.1 임베디드 보드

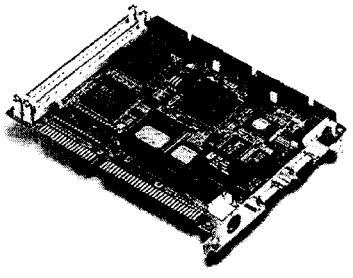

그림 3. MSC-410 임베디드 개발자 보드
Fig. 3 MSC-410 Embedded Development Board

개발하고자 하는 MSC-410 보드는 SVGA 컨트롤러와 PCI 이더넷 인터페이스 등이 하나의 보드에 내장되어 있는 486 보드이다. SVGA 컨트롤러는 1MB의 비디오 메모리를 가지며, CHIPS 65550 칩셋을 사용하고 2MB까지 확장 가능하다. 이 칩셋은 로컬 PCI 버스에서 사용되고 32 비트 그래픽 처리 용량을 가진다. 이것은 디스플레이 응용에 유리하며 TFT 등을 포함한 다양한 LCD 타입을 지원해 준다.

이더넷 Realtek RTL 8029AS PCI 버스 이더넷 컨트롤러로 인터넷으로의 접속을 가능하게하며 고속의 IDE 컨트롤러가 있어 하드디스크, CD-ROM 드라이브, 테이프 백업 드라이브 등의 연결이 가능하다.

또한, 고속의 RS-232 시리얼 포트와 플로피 드라이브도 지원한다. 그리고, 16KB의 L1 캐시 메모리와 128KB의 L2 캐시 메모리를 가지고 있다.

4.2 DiskOnChip

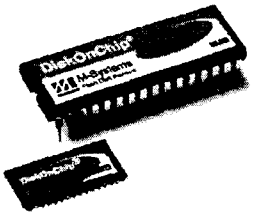

그림 4. DiskOnChip Millenium(8MB)
Fig. 4 DiskOnChip Millenium(8MB)

M-System 사[16]에서 출시되는 DOC는 일종의 하드 디스크 대용으로 사용할 수 있는 보드 내장형 플래시 디스크이다. 이 중 DOC Millenium은 DOC의 한 시리즈로서 하나의 다이(die)에 디스크 컨트롤러와 플래시 메모리를 조합한 형태이다. 또한 셋톱 박스, Thin Client, 웹폰, PDA, 자동차용 PC, Thin Server 그리고 단말기 등의 인터넷용 장치들을 만들기 쉽도록 최적화되어있다. 이러한 기기들은 최소의 무게, 크기, 전력소비 등을 필요로 하면서 대용량의 기억 장치를 제공해 주어야 한다. 이 DOC 계열의 플래시 디스크들은 이것을 만족시켜 주며 기존의 하드 디스크와 플로피 디스크를 대치할 수 있는 저가의 장치이다.

DOC의 응용분야를 살펴보면 다음과 같다.

- √ Embedded systems
- √ Internet access devices
- √ Internet set-top box, web WBT, thin client, Routers, networking
- √ Web-phones, car-PC, DVD, HPC
- √ Point of sale
- √ Industrial PC's
- √ Telecom
- √ Medical

4.3 개발 환경

이미 소개한바와 같이 하드웨어는 x86 계열의 AMD 486 프로세서를 내장한 MPC-410 개발자 보드와 8MB의 용량을 가진 DOC Millenium(MD-2800) 플래시 메모리이다. 기본적으로 개발자 보드에 DOC 소켓이 있기 때문에 플래시 메모리를 끼우기만 하면 된다.

추가적으로 48 배속의 CD-ROM 드라이브, 2GB의 하드 디스크 그리고 3.5" 플로피 드라이브를 개발 보드에 연결한다. 이것들은 개발환경 구축 및 포팅 그리고 플래시 메모리에서 부팅 가능하게 만드는데 도움을 준다. 환경 구축시 필요한 장치들을 정리하면 다음 표와 같다.

표 1. 개발 하드웨어 환경

App. Board	MPC-410 AMD 486 CPU
DiskOnChip	MD-2800 8MB Flash Memory
Memory	16MB RAM
FDD	3.5"
CD-ROM	LG 48X
HDD	2GB
Power Supply	5V 전원공급장치

그리고, 임베디드 시스템에서 DOC 구동을 위한 소프트웨어는 관련 홈페이지에서 구할 수 있는데, 필요한 파일 목록은 다음과 같다.

- √ dformat.exe: DOC 포맷 유틸리티
- √ dupdate.exe: DOC 펌웨어를 업데이트하기 위한 유틸리티
- √ docmap.exe: DOC 정보를 추출하기 위한 유틸리티
- √ doc42.exb: DOC 펌웨어 이미지
- √ doc2.fff: 또 다른 DOC 펌웨어 이미지

4.4 구현 과정

구현 과정은 복잡하며 여러 단계를 거친다. 특히, 커널 이미지 생성과 루트 파일 시스템을 구축하는데 시간이 소요된다. 전체적인 구현 단계들을 보면 다음과 같이 나눌 수 있다.

(1) Processes of the Real Time Linux Porting
- RTLinux Kernel 2.2.14

(2) Processes of the DOC RT Linux Porting

(3) Preparing for the DOC Bootable on the

Embedded Board
(4) Creating a Root File System
(5) Enabling Booting from the DOC
(6) Network Configuration on the DOC
(7) Making the Telnet Server on the DOC
(8) Installing and Setup the Minicom on the DOC
(9) Configuration Setup for User Login
(10) User Login from the Remote Terminal

기본적으로 개발보드에 연결되어 있는 하드 디스크에 RedHat 6.2, Kernel 2.2.14가 설치되어 있고, 이하 절에서 편의상 "bash#"는 개발환경 리눅스의 루트 권한 쉘 그리고 "doc@bash#"는 DOC로 루트 권한 로그인 했을 때 쉘을 나타낸다.

4.4.1 Processes of the Real Time Linux Porting

1) "bash#ftp ftp.rtlinux.com"하여 /pub/rtlinux/old/v2디렉토리에서 "rtlinux-2.2-prepatched.tar.gz"를 다운로드한다.
2) "rtlinux-2.2-prepatched.tar.gz"를 /usr/src 밑에 복사한다.
3) "bash#cd /usr/src"
4) "bash#gunzip rtlinux-2.2-prepatched.tar.gz"으로 압축을 풀면 rtlinux-2.2-prepatched.tar" 파일만 생긴다.
5) "bash#tar -xvf /usr/src/rtlinux-2.2-prepatched.tar"를 실행하여 압축을 풀면 "rtlinux-2.2"란 디렉토리가 생성된다.
6) /usr/src/linux에 연결된 링크 디렉토리를 기존 내용에서 /usr/src/rtlinux-2.2/linux로 연결 한다. 우선, "bash#rm /usr/src/linux"를 실행하여 기존 링크를 없앤다. 다시 "bash#ln -s /usr/src/rtlinux-2.2/linux /usr/src/linux"를 하여 새로운 링크를 만든다.
7) "bash#cd /usr/src/linux"
8) "bash#make menuconfig" 명령을 수행하여 환경을 설정한다.
   ■ [Code maturity level options] 항목을 "yes"로 설정
   ■ [Processor type and features] 항목은 "486 Processor family"로 설정
   [*] Symmetric multi-processing support를 설정한다.
   ■ Loadable module support 항목 중
   [*] Enable loadable module support 를 설정
   ■ General setup은 원래 설정 그대로
   ■ Plug and Play support 은 원래 설정 그대로
   ■ Networking options 은 원래 설정 그대로
   ■ SCSI support 은 원래 설정 그대로
   ■ Network device support 항목중
   [*] PCI NE2000
   ■ Kernel hacking항목중
   [*] Magic SysRq key를 설정
9) "bash#cd /usr/src/linux"를 실행한다.
10) "bash#make dep;make clean;make bzImage"을 실행하여 커널 이미지를 생성한다.
11) 생성된 이미지로 부팅을 테스트해 보기 위해 lilo를 다음과 같이 수정한다.

"bash#vi /etc/lilo.conf"

```
boot=/dev/hda
map=/boot/map
Install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.14
label=linux
root=/dev/hda1
read-only
image=/RTboot/vmRTlinuz-2.2.14    #추가한 부분
label=rtlinux  #추가한 부분
root=/dev/hda1  #추가한 부분
read-only  #추가한 부분
```

14) "bash#mkdir /RTboot"를 실행하여 디렉토리를 만든다.
15) "bash# cp /usr/src/rtlinux-2.2/linux/arch/i386/boot/bzImage /RTboot/vmRTlinuz-2.2.14"를 하여 커널이미지를 복사한다.
"lilo"를 실행하여 시스템에 적용시킨다.
16) "bash#lilo"를 실행한 후 재부팅 시킨다.
17) 부팅시 〈tab〉 키를 누르고 "linux boot:rtlinux"로 부팅한다.

4.4.2 Processes of the DOC RT Linux Porting

우선 rtlinux로 부팅한 상태에서 DOC 드라이버를 패치한다. 드라이버는 관련 홈페이지에서 구할 수 있는데 파일 이름은 "doclinux_2_2_x_1_21.zip"이다. 포팅 과정은 다음과 같다.

1) "bash#mkdir /tmp/temp"
2) 압축은 푼 후 드라이버 설치 파일 "driver.tgz" 를 /tmp/temp에 복사한다.
3) "bash#cd /tmp/temp"
4) "bash#tar -zxvf driver.tgz"을 실행하여 압축을 푼다.
5) 풀린 파일들을 제 위치에 복사하기 위하여,
   "bash#cd /tmp/temp"
   "bash#rm driver.tgz"
   "bash#cp -rf . /"
   "bash#cd /tmp"
   "rm -fr temp"
6) 다음은 수동으로 커널을 패치해야 하는데 패치해야 할 파일들은 다음과 같다.
   /usr/src/linux/include/linux/blk.h
   /usr/src/linux/include/linux/major.h
   /usr/src/linux/drivers/block/Makefile
   /usr/src/linux/drivers/block/Config.in
   /usr/src/linux/drivers/block/ll_rw_blk.c
   상기 파일들을 다음과 같이 수정한다.
7) if ["$CONFIG_BLK_DEV_RAM" = "y"]
   then bool Initial RAM disk (initrd) support'
   CONFIG_BLK_DEV_INITRD
   fi
   +
   +bool 'M-Systems DiskOnChip'
   CONFIG_BLK_DEV_GENERIC_FLASH_DOC
   +
   tristate 'XT harddisk support'
   CONFIG_BLK_DEV_XD.
   여기서 "+" 표시된 줄을 추가한다.
8) "bash#vi /usr/src/linux/drivers/block/Makefile" 후 다음과 같이 파일을 편집한다.
   +ifeq
   ($(CONFIG_BLK_DEV_GENERIC_FLASH_DOC),y)
   +L_OBJS += flash_doc/fl.o
   +endif
   +
   ifeq ($(CONFIG_BLK_DEV_LOOP),y)
     LX_OBJS += loop.o
   else
   여기서 "+" 표시된 줄을 추가한다.
9) "bash#vi /usr/src/linux/include/linux/major.h" 후 다음과 같이 파일을 편집한다.
   #define IDE5_MAJOR 57
   +#define IGEL_FLASH_MAJOR 62
   #define SCSI_DISK1_MAJOR 65
   #define SCSI_DISK2_MAJOR 66
   여기서 "+" 표시된 줄을 추가한다.
10) "bash#vi /usr/src/linux/drivers/block/ll_rw_blk.c" 파일을 다음과 같이 편집한다.
   #endif CONFIG_BLK_DEV_MD
   +#ifdef
   CONFIG_BLK_DEV_GENERIC_FLASH_DOC
   +fl_init()
   +#endif
   CONFIG_BLK_DEV_GENERIC_FLASH_DOC
   여기서 "+" 표시된 줄을 추가한다.
11) "bash#vi /usr/src/linux/include/linux/blk.h" 후 다음과 같이 파일을 편집한다.
   extern int md_init(void)
   +extern int fl_init(void)
   +
   ~
   #define DEVICE_NAME "Sanyo H94A CD-ROM"
   #define DEVICE_REQUEST do_sjcd_request
   +#define DEVICE_NR(device) (MINOR(device))
   +#define DEVICE_ON(device)
   +#define DEVICE_OFF(device)
   +
   +#elif (MAJOR_NR == IGEL_FLASH_MAJOR)
   +
   +#define DEVICE_NAME "IGEL Flash Disk"
   +#ifdef CONFIG_BLK_DEV_GENERIC_FLASH_DOC
   +#define DEVICE_REQUEST fl_request
   +#else
   +#define DEVICE_REQUEST igflash_request
   +#endif
   +#define DEVICE_NR(device) (MINOR(device))
   +#define DEVICE_ON(device)
   +#define DEVICE_OFF(device)
   +
   +#elif (MAJOR_NR ==

```
IGEL_SWAP_MAJOR)
+
+#define DEVICE_NAME "IGEL Swap
 Device"
+#define DEVICE_REQUEST
 igswap_request
 #define DEVICE_NR(device)
 (MINOR(device))
```

여기서 "+" 표시된 줄을 추가한다.

12) "bash#cd /usr/src/linux"
13) "bash#make menuconfig"를 실행하여 환경을 재설정 한다.
    ■ [Block devices] 항목에
      [*] M-Systems DiskOnChip (NEW)
14) "bash#make clean"
15) 커널에 DOC을 추가 하기 위해,
    "bash#cd /usr/src/linux/drivers/block /flash_doc"
16) "bash#make"
17) "bash#cd /usr/src/linux"
18) "bash#make dep;make bzImage"를 실행하여 커널이미지를 재생성한다.
20) "bash#cd /"
21) "bash#mkdir DOCRTboot"
22) "bash#cp /usr/src/rtlinux-2.2/linux/arch/i386/boot/bzImage/DOCRTboot/vmRTlinuz-2.2.14"
23) "bash#/etc/lilo.conf" 파일에 다음과 같이 lilo를 추가한다.

```
boot=/dev/hda1
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.14
  label=linux
  root=/dev/hda1
  read-only
image=/RTboot/vmRTlinuz-2.2.14
  label=rtlinux
  root=/dev/hda1
  read-only
image=/DOCRTboot/vmRTlinuz-2.2.14# 추가
  label=docrtlinux # 추가
  root=/dev/hda1 # 추가
  read-only # 추가
```

    "lilo"를 실행하여 시스템에 적용시킨다.
24) "bash#lilo"를 실행한 후 재부팅시킨다.
25) 부팅시 <tab> 키를 누르고 "linux boot: docrtlinux"로 부팅한다.
26) "bash#cd /usr/src/linux"
27) "bash#make modules;make modules_install"를 실행한다.

4.4.3 Preparing for the DOC Bootable on the Embedded Board

실시간 커널에 DOC를 패치하여 추가하였으면 우선 DiskOnChip을 위하여 inode 들을 만들어야 한다.

1) "bash#cd dev"
2) 다음과 같이 실행하여 디바이스를 만든다.
   "bash#mknod fla b 62 0"
   "bash#mknod fla1 b 62 1"
   "bash#mknod fla2 b 62 2"
   "bash#mknod fla3 b 62 3"
   "bash#mknod fla4 b 62 4"
3) 이제 DOC의 펌웨어를 업데이트 하기 위해 DOS로 재부팅한다.
4) 디스켓에는 앞절에서 소개한 유틸리티들이 들어 있어야 한다.
5) "a:dformat /win:d400 /s:doc2.fff /y
   옵션에서 /win 는 DiskOnChip 의 데이터를 엑세스할 때 사용하는 메모리 윈도우 지정 어드레스이다 .만약 실행 에러가 난다면 이 값을 다른 값을 사용하면 된다.
6) DOC의 파티션을 잡기 위하여 "docrtlinux"로 재부팅한다.
7) "bash#fdisk /dev/fla" 후 다음의 과정을 수행한다.
   ■ 기존의 파티션 내용을 확인한다.
     Command(m for help): p
   ■ 기존의 파티션을 지운다.
     Command(m for help): d
     Partition number(1-4): 1
   ■ 새로운 파티션을 만든다.
     Command(m for help): n
     Command action extended primary partition (1-4) p
     Partition number (1-4): 1
     First cylinder (1-XXX): 1
     Last cylinder or +size or +sizeM or +sizeK ([1] XXX):XXX
   ■ 파티션 타입을 Linux native로 바꾼다.
     Command (m for help): t
     Partition number (1-4): 1
     Hex code (type L to list codes): 83
   ■ 파티션을 부팅 가능하게 설정한다.
     Command (m for help): a
     Partition number (1-4): 1
   ■ 제대로 파티션이 설정되었는지 확인한다.
     Command (m for help): p
   ■ 파티션 테이블을 저장한다.

Command (m for help): w

8) 파티션을 잡았으므로 "docrtlinux"로 재부팅한다.
9) "bash#mke2fs /dev/fla1"으로 EXT2 포맷하면 다음과 같은 메시지가 나타난다.
mke2fs 1.14, 9-Jan-2002 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
1992 inodes, 7959 blocks
397 blocks (4.99%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
1992 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
이젠 리눅스에서 사용가능하므로 마운트시켜 본다.
10) "bash#mkdir /diskonchip"
11) "bash#mount /dev/fla1 /diskonchip"
12) "bash#cd /diskonchip"
13) "bash#ls"하면 아무 것도 나타나지 않지만 이제 준비가 된 상태이다.

4.4.4 Creating a Root File System

8MB의 작은 공간에 필요한 파일들을 넣는다는 것은 상당히 번거롭고 시간을 많이 소비하는 작업이다. "docrtlinux"로 부팅한 상태에서 루트 파일 시스템은 다음의 순서로 구축하였다. 아래의 파일들에서 루트 파일 시스템에서 꼭 필요로 하지 않는 파일들도 있으나 이는 DOC의 공간이 허용하는 한 관리의 편의를 위해 복사해 놓은 파일들이다.

1) "bash#cd /diskonchip"
2) "bash#mkdir bin boot dev etc home lib mnt proc root usr var"
3) "bash#ln -s bin sbin"을 하여 모든 실행 파일들을 bin에 저장시키도록 한다.
4) 각 디렉토리에 필요한 최소한의 파일들을 복사한다.
5) 파일들을 복사할 때 참고할 사항들은 다음과 같다.
"bash#cp -dpR /dev/ttyS? /diskonchip"
"bash#cp -dr /etc/rc.d /diskonchip/etc"
"bash#cp -d /etc/inittab /diskonchip/etc"
"bash#cp /etc/passwd /diskonchip/etc"
6) 파일들을 복사할 때 실행 파일들의 의존성 관계를 조사하여 필요한 라이브러리를 복사해야 한다.
예를 들어, "bash#ldd ls" 하면 다음과 같은 필요한 라이브러리들이 도출된다.
libtermcap.so.2 => /lib/libtermcap.so.2
libc.so.6 => /lib/libc.so.6
/lib/ld-linux.so.2 => /lib/ld-linux.so.2
그 다음에 다음과 같이 위 라이브러리들을 복사한다.
"bash#cp /lib/libtermcap.so.2 /diskonchip/lib"
/lib/ld-linux.so.2는 링크를 형성하고 있으므로 복사할 때 옵션을 주어야 한다.
7) 위에서 파일들을 복사할 때 라이브러리들은 파일의 크기가 크다. 이를 줄이기 위한 방법은 다음과 같다.
"bash#objcopy -strip-debug /diskonchip/lib/라이브러리이름"

4.4.5 Enabling Booting from the DOC

루트 파일 시스템을 구축했으면 DOC로부터 부팅 가능한 설정들을 해 주어야 한다.

1) "bash#vi /diskonchip/etc/fstab"을 하여 다음과 같이 편집한다.
"/dev/fla1 / ext2 defaults 1 1
/proc /proc proc defaults 0 0"
2) "bash#chroot /diskonchip /sbin/ldconfig"
3) "bash#mkdir /diskonchip/boot"
4) "bash#cp /usr/src/linux/arch/i386/boot/bzImage /diskonchip/boot/doc2000"
5) "bash#rdev /diskonchip/boot/doc2000 /dev/fla1"
6) "bash#cp /tmp/doc-driver/plilo /diskonchip/sbin"
7) "bash#cp/tmp/doc-driver/boot.b /diskonchip/boot"
8) "bash#cp/tmp/doc-driver/lilo.conf /diskonchip/etc"
9) "bash#/diskonchip/sbin/plilo-C /diskonchip/etc
/lilo.conf -i /diskonchip/boot/boot.b -m/diskonchip/boot/map"
10) "bash#cd /"
11) "bash#umount /dev/fla1"
12) "bash#reboot" 후 유틸리티가 들어 있는 디스켓을 넣고 디스켓으로 부팅한다.
13) "a:dupdate /win:D400 /s:DOC42.exb"
14) 재부팅 후 BIOS에서 하드 디스크 설정를 없애고 부팅한다.
올바르게 되었으면 DOC로 부팅된다. DOC로 부팅시 에러가 있는지를 살펴보고 에러가 있으면 수정해야 한다.

4.4.6 Network Configuration on the DOC

DOC로 부팅 후 네트워크를 설정한다.

1) "doc@bash#vi /etc/sysconfig/network-scripts/ifcfg-eth0" 후 다음과 같이 입력한다.

"DEVICE="eth0"
BOOTPROTO="none"
IPADDR="203.246.99.10"#맞는 주소로 변경
NETMASK="255.255.255.0" #맞는 주소로 변경
ONBOOT="yes""

2) "doc@bash#vi /etc/sysconfig/network" 후 다음과 같이 입력한다.
"NETWORKING=yes
HOSTNAME="cjk" #원하는 이름으로
GATEWAY="203.246.99.1"" #맞는 주소로 변경
3) 재 부팅 한다.

4.4.7 Making the Telnet Server on the DOC

루트 파일 시스템 구성시 /usr/bin/telnet, /usr/sbin/inetd 등의 실행 파일을 복사하고 다음과 같이 설정 파일을 변경한다.

1) "doc@bash#vi /etc/inetd.conf" 후 아래와 같이 설정 한다.
"telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd"
재 부팅하여 텔넷 데몬이 잘 작동하는지 체크한다.

4.4.8 Installing and Setup the Minicom on the DOC

미니콤은 리눅스에 있는 시리얼 통신 프로그램으로 초기에 설정을 해 주어야 사용할 수 있다. 그리고, 당연히 미니콤 프로그램이 루트 파일 시스템 구축시 함께 구축되어야 한다.

1) "doc@bash#minicom -s"
{configuration}
Filenames and paths
File transfer protocols
Serial port setup
Modem and dialing
Screen and keyboard
Save setup as dfl
Save setup as..
Exit
2) 위와 같은 화면에서 "Serial port setup"을 선택하여 다음과 같이 설정을 맞춘다.
A - Serial Device : /dev/ttyS0
B - Lockfile Location : /var/lock
C - Callin Program :
D - Callout Program :
E - Baud/Par/Bits : 9600 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No
3) "doc@bash#minicom"을 실행하여 본다.

4.4.9 Configuration Setup for User Login

원격지에서 텔넷으로 로그인 후 곧바로 시리얼 통신 프로그램이 실행되도록 설정하면 된다. 원래는 "adduser" 명령으로 사용자 계정을 만들어 주어야 하나 DOC의 공간 문제로 모체가 되는 하드 디스크에서 DOC를 마운트 한 후 사용자 계정을 만들어 주고 /etc/passwd, /etc/shadow 파일을 DOC의 /etc에 복사하였다. 그 과정은 다음과 같다.

우선 BIOS에서 하드 디스크를 다시 살린 후 하드디스크의 "docrtlinux"로 부팅한다.

1) "bash#mount /dev/fla1 /diskonchip"
2) "bash#adduser yoojin" 하면 /home/yoojin이 만들어지고 /etc/passwd, /etc/shadow에도 만들어진다.
3) "bash#passwd yoojin"
"Changing password for user yoojin
New UNIX password : xxxxxx #패스워드 입력
Retype new UNIX password : xxxxxx #동일한 패스워드 입력
4) "bash#mkdir /diskonchip/home/yoojin"
5) "bash#cp /etc/passwd /diskonchip/etc"
6) "bash#cp /etc/shadow /diskonchip/etc"
7) 재부팅 후 BIOS에서 하드 디스크를 지우고 DOC로 부팅한다.
8) "doc@bash#vi .bashrc"로 로그인시 아래와 같이 환경을 설정한다.

```
# .bashrc
# User specific aliases and functions
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias ls='ls -aCF'
minicom# 추가
# Source global definitions
if [ -f /etc/bashrc ] then
   . /etc/bashrc
fi
```

4.4.10 User Login from the Remote Terminal

다른 컴퓨터에서 텔넷으로 사용자 계정으로 로그인 하여 시리얼 통신 프로그램이 구동되는지 테스트한다. 테스트 화면은 다음 그림 5, 그림 6, 그림 7과 같다.

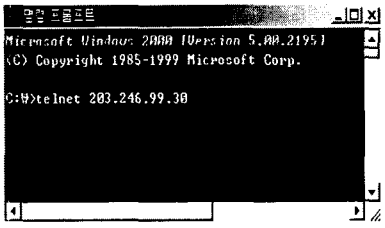

그림 5. 테스트 화면(1)
Fig. 5 Captured Test Screen(1)

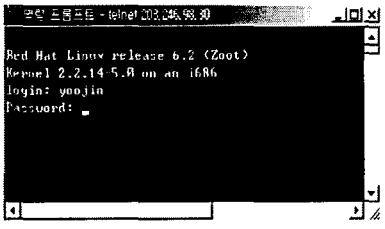

그림 6. 테스트 화면(2)
Fig. 6 Captured Test Screen(2)

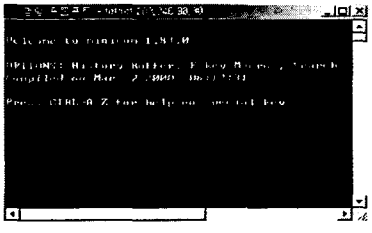

그림 7. 테스트 화면(3)
Fig. 7 Captured Test Screen(3)

V. 연구 결과

개발 완료된 보드를 가지고 시리얼 교환기를 가지고 있는 회사에서 필드 테스트를 진행하였다. 우선 보드에서 부팅 후 회사의 IP 주소와 넷마스크 그리고 게이트웨이를 설정한다.

1) "doc@bash#vi /etc/sysconfig/ifcfg-eth0" 후 다음과 같이 입력한다.
"DEVICE="eth0"
BOOTPROTO="none"
IPADDR="61.72.47.XX"
NETMASK="255.255.255.0"
ONBOOT="yes""
2) "doc@bash#vi /etc/sysconfig/network" 후 다음과 같이 입력한다.
"NETWORKING=yes
HOSTNAME="cjk"
GATEWAY="203.246.XX.X""
3) 보드를 셧다운시키고 전원을 끈 후 교환기의 시리얼 케이블을 보드에 연결한다.
4) 이더넷 케이블을 연결한다.
5) DOC로 부팅한다.

다음은 최종적으로 개발 완료한 화면이다. 그림 8은 원격지에서 "Tera Term Pro"라는 시리얼 통신 프로그램을 이용하여 접속한 초기 화면이다. 그리고 그림 9는 원격지에서 개발된 보드에 접속하기 위해 사용자 계정으로 로그인하는 화면이다.

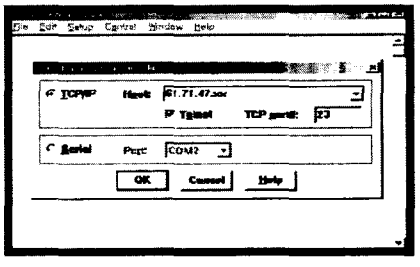

그림 8. 개발 완료 화면(1)
Fig. 8 Completed Screen(1)

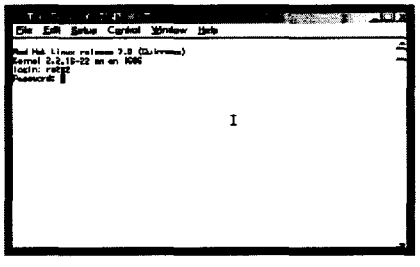

그림 9. 개발 완료 화면(2)
Fig. 9 Completed Screen(2)

그림 10은 실시간으로 전해오는 시리얼 교환기의 상태를 점검하기 위해 디바이스를 초기화하는 화면이다.

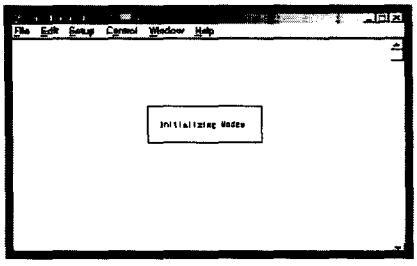

그림 10. 개발 완료 화면(3)
Fig. 10 Completed Screen(3)

그림 11과 그림 12는 각각 교환기 상태를 실시간적으로 보여주고 있는 화면이며 실시간으로 제어도 가능함을 알 수 있다.

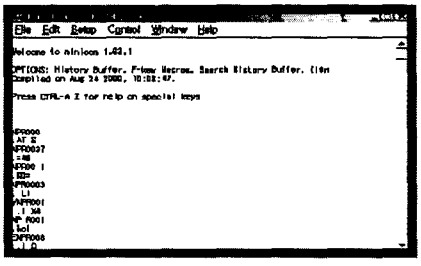

그림 11. 개발 완료 화면(4)
Fig. 11 Completed Screen(4)

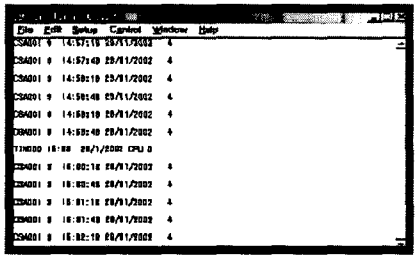

그림 12. 개발 완료 화면(5)
Fig. 12 Completed Screen(5)

VI. 결론

지금까지 실시간 운영체제를 이용한 임베디드 시스템으로서 STEP 컨버터 개발 과정을 소개하였고, 개발 완료한 결과를 보여주었다.

486 임베디드 보드에 DOC 플래시 메모리를 탑재하고, 여기에 실시간 리눅스 운영체제 커널을 포팅하여 부팅 가능하도록 만들었다. 그리고, 루트 파일 시스템을 구축하여 각종 명령어들과 필요한 라이브러리들을 설치하였다. 적은 용량의 메모리에 이들을 설치해야하므로 최소한의 커널과 루트 파일 시스템을 적재하도록해야 했다. 그리고 시리얼 통신 프로그램인 미니콤을 설치하여 원격지에서도 접속이 가능하도록 하였다. 일련의 과정들을 거쳐 마지막 단계까지 성공적으로 개발이 완료되었음을 확인하였다.

개발한 임베디드 시스템은 교환기뿐만 아니라 각종 시리얼 장비들에 유용하게 이용될 수 있다.

향후에는 각종 응용 프로그램을 개발하여 더욱 편리하게 임베디드 시스템 환경을 구축할 예정이며, 개발 경험을 바탕으로 최근의 이슈가되고 있는 무선 단말기 분야에 응용해 보고자 한다.

참고문헌

[1] 주민규외 4, "내장형 리눅스를 이용한 라우터의 설계 및 구현," 한국정보처리학회 논문집(A), pp. 339~ pp. 344, 2001. 12.

[2] M-Systems, "Using the DiskOnChip with Linux OS," IM-DOC-21, 1999, 4.

[3] Victor Yodaiken, "The RTLinux Manifesto", http://www.rtlinux.org

[4] M. Barabanov and V. yodaiken,"Real-Time Linux", Linux journal, 1997, 2.

[5] Jerry Epplin , "Linux as an Embedded Operating System", http://www.linuxfocus.org

[6] Ismael Ripoll, "Real Time Linux I,II,III", http://www.linuxfocus.org

[7] Jerry Epplin, "Linux as an Embedded Operating System", http://www.espmag.com/97/fe39710.htm

[8] Matt Welsh, "Implementing Loadable Kernel Modules for Linux", http://www.ddj.com/ddj/1995/1995.05/welsh.html
[9] R Magnus, U Kunitz, M Dziadzka, D Verworner M Beck, H Bohme " Linux Kernel Internals ", 도서출판 에프원, 2001.
[10] Seagull723 Preliminary Data Book, C&S Technology, http://www.cnstec.com
[11] John Lombardo, "Embedded Linux", New Riders, 2001, 6.
[12] Moshe Bar, "Linux File Systems", McGraw Hill, 2001.
[13] http://kldp.org
[14] http://www.kesl.org
[15] http://kelp.or.kr
[16] http://www.m-sys.com
[17] http://www.flashdisk.co.kr

저 자 소 개

최 준 기
1993. 2. 순천향대학교 전산학과(공학사)
1995. 2. 순천향대학교 전산학과(공학석사)
1999. 2. 순천향대학교 전산학과(공학박사)
1999. 3. ~ 현재 인덕대학 여성정보행정과 조교수