# Table of Contents

Part I. EOS uClinux Software Packages ............................................................................. 4  
Part II. How to Use EOS uClinux .......................................................................................... 5  
1 Where to Find EOS uClinux Kernel and Application Programs ........................................ 5  
2 How to Compile EOS uClinux ............................................................................................ 6  
3 How to Compile User Programs and Prepare the RAM Disk .................................................. 8  
4 How to run uClinux on the EOS demo Board (Nand Flash Boot Method) ............................. 12  
5 User Defined Setting ............................................................................................................ 15  
  5.1 IP Address ...................................................................................................................... 15  
  5.2 Ethernet MAC Address .................................................................................................. 15  
6 How to use the Bootloader .................................................................................................. 17  
  6.1 Introduction .................................................................................................................... 17  
  6.2 Functionality of the Bootloader ...................................................................................... 17  
  6.3 Instruction Sets ............................................................................................................... 17  
  6.4 Setting up “bootp” and “tftp” for a Host PC (a Linux Server) ........................................... 17  
  6.5 How to Use Bootp, tftp .................................................................................................. 20  
  6.6 Serial Downloading ....................................................................................................... 21  
  6.7 Writing and Erasing the Nand Flash ............................................................................. 22  
Part III. How to use application programs ........................................................................... 25  
7 Busybox Utility .................................................................................................................. 25  
  7.1 Ping .................................................................................................................................. 25  
  7.2 Telnet .............................................................................................................................. 25  
  7.3 Tftp .................................................................................................................................. 26  
  7.4 Wget ................................................................................................................................ 26  
8 An Example for a Keypad and a device driver Keypad .......................................................... 27  
9 Example of RTC ................................................................................................................ 28  
10 NFS (Network File System) ................................................................................................. 30  
  10.1 Configuring the kernel ................................................................................................... 30  
  10.2 Setting up the NFS (Server Linux) ............................................................................... 30  
  10.3 Running portmapper (Server Linux) ............................................................................. 31  
  10.4 Setting up a NFS server (setting up a host) ................................................................. 31  
  10.5 Setting up a NFS client (Setting up a target board) ...................................................... 33  
  10.6 Mount NFS using the shell script ................................................................................. 33  
  10.7 NFS Test ....................................................................................................................... 33
List of Figures

Figure 2-1: Copy process of the kernel and RAM disk images. ......................... 7
Figure 4-1: The mapping between the CFGR and board jumper. ...................... 12
Figure 4-2: Auto boot mode (uClinux boot.) ............................................... 13
Figure 4-3: Auto boot mode (bootlader boot.) ............................................. 13
Figure 4-4: Normal boot mode (boot from rom). ......................................... 13
Figure 4-5: Nand Flash Address Map......................................................... 14
Figure 5-1: Examples of the “rc2” and “resolv.conf” stored in the Nand flash ... 15
Figure 5-2: View mac address stored in the nand flash ................................ 16
Figure 5-3: Setting mac address .................................................................... 16
Figure 6-1: List of bootloader command ...................................................... 17
Figure 6-2: An example of the bootp. ........................................................... 20
Figure 6-3: An example of the tftp. .............................................................. 21
Figure 6-4: An example of the serial data transfer in the bootloader ............... 22
Figure 6-5: An example of transferring the serial data from the bootloader. ... 23
Figure 7-1: An example of using the ‘ping’ program .................................... 25
Figure 7-2: An example of the telnet program. ............................................ 25
Figure 7-3: An example of using the ‘tftp’ program ..................................... 26
Figure 7-4: An example of using the ‘wget’ program ................................... 26
Figure 8-1: A definition of a keypad driver Keypad. .................................... 28
Figure 9-1: Example of the RTC ................................................................. 29
Part I. EOS uClinux Software Packages

<table>
<thead>
<tr>
<th>Linux Kernel</th>
<th>2.4.20 (patched to uClinux)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolchain</td>
<td>GNU Cross Compiler, elf2flt</td>
</tr>
<tr>
<td>Library</td>
<td>uClibc-0.9.18</td>
</tr>
<tr>
<td>Bootloader</td>
<td>Support Bootp, Tftp Downloading, Nand flash Programing</td>
</tr>
</tbody>
</table>

**Device Driver**
- Serial driver (can support uart based modem)
- Keypad Driver (EOS Embedded keypad)
- STN LCD Driver
- MTD Flash driver (NAND:YAFFS, NOR:JFFS),
- Ethernet Device driver (CS8900)
- RTC (DS1302(GPIO), M41T00(IIC))
- Magnetic Card reader (GSR-1160)
- PWM, ADC
- EEPROM(AT24CXX(IIC))
- * Including all device driver test program

**Support Network**
- TCP/IP, UDP, FTP, DHCP, ARP, ICMP, IGMP, NFS, NAT, PPP

**User Utility**
- Busybox-0.60.5
- Tinylogin-1.4
- PPPD, PPPoE, DHCPD, DHCPCD

---

![Diagram](https://via.placeholder.com/150)

*그림 0-1 EOS uClinux hardware/software spec.*
Part II. How to Use EOS uClinux

1 Where to Find EOS uClinux Kernel and Application Programs

The files can be obtained from the CD provided, or by downloading the following files from the ftp site: 
ftp://ftp.adc.co.kr/pub/uclinux/EOS-uClinux

- EOS-uClinux-2.4.20-2004xxxx.tar.gz: EOS uClinux kernel source
- eos_bootloader_rom_xxxx.tar.gz: bootloader executed in the Rom
- eos_bootloader_nand_xxxx.tar.gz: bootloader stored in the Nand and executed in the Ram
- ae32000-elf2flt.tar.gz: Convert ELF format to binflat format
- uClibc-0.9.18-ae32000.tar.gz: application program library
- busybox-0.60.5-ae32000.tar.gz: Linux application program
- tinylogin-1.4-ae32000.tar.gz: Simple login program
- eos_uclinux_nand_boot_xxxx.zip: Nand flash auto boot program
- bootp_setup.tar.gz: Linux bootp, tftp server program
- Additional programs (keypad_test.tar.gz, lcd_test.tar.gz, rtc_test.tar.gz sig_test-ae32000.tar.gz, uart0_test.tar.gz, ...)

After you have obtained the files above, you should uncompress all the files. Then unpack the files using the “tar” command. For example: tar -xfz [filename]. Afterwards, type a “tar” command. For example: tar xzf [a filename].

Afterwards, using the "ls" command, you should see the following directories:

```
> ls
ac32000-elf2flt
busybox-0.60.5-ae32000
tinylogin-1.4-ae32000
uClibc-0.9.18-ae32000
EOS-uClinux-2.4.20-2004xxxx
...
```
2 How to Compile EOS uClinux

To compile the kernel, you need to reconfigure it, check for dependencies, and then clean up files. We recommend you use the default configuration options given when you compile. The default configuration options are stored in the file “arch/ae32000bnommu/def-configs.” If you want to select a different CPU, device, or another configuration option, you must run “make” with the “menuconfig” option and save that. It will create “.config” in the kernel directory. If you want to use what you created in the future, store it in “arch/ae32000bnommu/def-configs” with a name that you desire (for example, eos). To finish reconfiguring your kernel, run ‘make’ with ‘_config’ attached to the file name of the file you just created (for example, eos_config).

- Make eos_config

Run the following commands. The following process generates a “linux.bin” file.

- Make oldconfig  # Configures the system.
- Make dep         # Checks for dependencies
- Make             # Compiles the kernel

To run Linux kernel, you need the Root File System. See next section for root file system. Also, to run Linux kernel, you need bootloader. The bootloader initializes basic hardware first, and then copies the Linux kernel and RAM disk images that are stored in the FLASH region into the SDRAM region. Finally, it makes the program control jump to the kernel starting point. Bootloader copy the kernel image to 0xC000_0000 and ramdisk image to 0xC050_0000 of sram.

You can download the kernel image and ramdisk image to SDRAM region by using “Sdownloader” or “tftp.”, and store these images in the Nand flash. See section 5 for details.

In order for the bootloader to copy the Linux kernel and RAM disk images into SDRAM, it must know the starting address in flash, starting address in SDRAM, and size for both the Linux kernel and RAM disk image. If the size of the Linux kernel image changes after it has been reconfigured, you must change the value of the linux image size in the bootloader or nand boot code to reflect its new size. It is not recommended for the user to change the starting address of the Linux kernel image in SDRAM. The starting location of the RAM disk image is defined in the “eos_bootloader/include/config.h” in the bootloader.

The “linux.bin” file is set to execute instruction in the address, “0xC0000000,” in SDRAM. You must check the definition of ‘START ADDRESS’ in the Linux kernel source in “arch/ae32000bnommu/vmlinux-ae32000.lds.in”. The following code shows the definition of ‘START ADDRESS:

```c
OUTPUT_ARCH(ae32000)
ENTRY(_stext)
SECTIONS
{
    . = 0xC0000000; /* START ADDRESS of Kernel */
```
The starting address and size of the RAM disk image in the kernel is defined in "arch/ae32000bnommu/mach-eos/arch.c". Here is an example of those definitions:

```c
static void __init
    fixup_eos(struct machine_desc *desc, struct param_struct *params,
              char **cmdline, struct meminfo *mi)
{
    ROOT_DEV = MKDEV(RAMDISK_MAJOR,0);
    setup_ramdisk(1, 0, 0, 2*1024);
    setup_initrd(0xC0500000, 1024*1024); // the starting address and the size of the RAM disk image
}
```

Figure 2-1 shows data area of bootloader.

<table>
<thead>
<tr>
<th>Bootloader Ram Data</th>
<th>Interrupt Vector Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC0000000</td>
<td>0xC0FFFC00</td>
</tr>
<tr>
<td>0xC0500000</td>
<td>0xC0FFFF0</td>
</tr>
<tr>
<td>0xC0700000</td>
<td>0xC0FFFFFF</td>
</tr>
<tr>
<td>SDRAM(16MB)</td>
<td></td>
</tr>
<tr>
<td>Kernel Image</td>
<td>0xC0000000</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Ramdisk Image</td>
<td>0xC0500000</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2-1:** Copy process of the kernel and RAM disk images.
3 How to Compile User Programs and Prepare the RAM Disk

- To build an application program for EOS embedded uClinux, you must add header and library (uClibc) files for Linux inside a ‘Makefile’ when you compile. Even though the compiler has this information when it is installed, the compiler does not include them but gets them from a ‘Makefile’ since it can be run under a NON-OS environment.
- You should include the directory path for the uClibc library.
- While linking by using “se3208-elf-ld”, you must include a linker script file that defines a starting address for a program and so on. You can do it by including “elf2flt.ld” as the linker option (the “elf2flt.ld” is provided by the “elf2flt.”)
- We will explain other options by looking through the “Makefile” for the “init” program.

```
CROSS_COMPILE  = ae32000-elf-
AS  = $(CROSS_COMPILE)as
LD  = $(CROSS_COMPILE)ld
CC  = $(CROSS_COMPILE)gcc
CPP  = $(CC) -E
AR  = $(CROSS_COMPILE)ar
NM  = $(CROSS_COMPILE)nm
STRIP  = $(CROSS_COMPILE)strip
OBJCOPY  = $(CROSS_COMPILE)objcopy
OBJDUMP  = $(CROSS_COMPILE)objdump
RANLIB  = $(CROSS_COMPILE)ranlib

AE32000_VER=ae32000-20030710
GCCDIR=$(shell pwd)/../../../../local/ae32000-elf-tools/lib/gcc-lib/ae32000-elf/egcs-2.91.66
GCCINC=$(GCCDIR)/include
UCLIBCDIR=$(shell pwd)/../uClibc-0.9.18-$(AE32000_VER)
UCLIBCINC=$(UCLIBCDIR)/include
LINUXDIR=$(shell pwd)/../uClinux-2.4.20-ae32000b
LINUXINC=$(LINUXDIR)/include
INCDIR = -I$(UCLIBCINC) -I$(GCCINC) -I$(LINUXINC) # ------- (1)
CFLAGS = -O2 -g -Dlinux -D__linux__ -Dunix -D__uclinux__ -DEMBED -nostdinc $(INCDIR)  # -----(2)
LDFLAGS = -r -Xlinker -Tae32000-elf2flt.ld -nostdfiles -nostdlib  # ---------- (3)
LIBDIR = $(shell pwd)/..uClibc-0.9.18-$(AE32000_VER)/lib
LIBS = -lm -lc
LDLIBS = -L$(LIBDIR) $(LIBS)

.c.o:
  $(CC) $(CFLAGS) -c $< -o $@
.s.o:
  $(AS) $< -o $@
```
export AS LD CC CPP AR NM STRIP OBJCOPY OBJDUMP RANLIB CFLAGS

CRTOBJ = $(LIBDIR)/crt0.o
INIT = init.elf
INITOBJ = simpleinit.o debug-ac32000b.o

all: $(INIT)

$(INIT): $(INITOBJ) debug-ae32000b.o pathnames.h
    $(CC) $(LDFLAGS) -o $@ $(CRTOBJ) $(INITOBJ) -lgcc $(LDLIBS)
    $(OBJDUMP) -DS $@ > $@.dis
    $(OBJDUMP) -x $@ > $@.x.dis
    ../ae32000-elf2flt/elf2flt $@

clean:
    rm -f simpleinit $(EXEC1) $(EXEC2) *.elf *.gdb *.o *.dis

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INCDIR: you must set a path properly for header files such as uClibc and uClinux for the compiler.</td>
</tr>
<tr>
<td>2</td>
<td>Options for the C compiler.</td>
</tr>
<tr>
<td></td>
<td>-nostdinc option in (2): sets not to include the newlib in the cross compiler into the header file.</td>
</tr>
<tr>
<td></td>
<td>-D option in (2): symbol definition</td>
</tr>
<tr>
<td></td>
<td>(3): linker option</td>
</tr>
<tr>
<td></td>
<td>-r option in (3): contains relocation information for a ELF file. In order to execute the ‘elf2flt’ command, it is required to set. However, it is hard to debug a disassemble file generated with -r option since it does not contain addresses for symbols which should be relocated. Therefore, if you want to see a disassemble file, you must compile it without -r option.</td>
</tr>
<tr>
<td></td>
<td>After the ELF file is created, the “elf2flt” command defined in (4) will be executed to generate an executable file in the FLAT format, which can be executed in the uClinux. The executable file generated is named “init.elf.bflt.”</td>
</tr>
<tr>
<td></td>
<td>If you want to compile other application programs by yourself, you must make a copy of the above makefile, add names of files that you want to compile, and then run the “make” command. If it produces some error messages during compilation, you have to check the paths for “include” and “uClibc” files again.</td>
</tr>
<tr>
<td></td>
<td>You can compile user programs by using the “make” command. You must set the path of each directory correctly for the Linux kernel, uClibc and compiler in the “Makefile” to be used by the “make” command. To remove object files that you created earlier, you must run the ‘make’ command with the ‘clean’ option (i.e. “make clean”).</td>
</tr>
<tr>
<td></td>
<td>Outputs from program compile are executable files for Linux such as “busybox.elf”. If you want to run the “elf” programs in uClinux, you must convert them into a flat format. The size of the executable files in flat format is smaller</td>
</tr>
</tbody>
</table>
than those in “elf.” The header and relocation information is simpler in flat format than in “elf” format. You can convert files in “elf” format into flat format by using the “elf2flt” command in the “ae32000-elf2flt” folder (since this process is run inside the “Makefile”, you need to set a path of the “elf2flt” executable file appropriately.) It will produce a file with a “bflt” extension (for example, “busybox.elf.bflt”). You have to rename and store the binflat executable file inside the RAM disk. Keep in mind that you operate the “elf2flt” command only in Linux.

“uClibc” is pre-compiled by developers of the kernel, so it is tailored to the environment. It is should not be recompiled by using “make clean” or “make all”. The uClibc library is stored in the “lib” directory. You must store and compile application programs in this “lib” directory.

The RAM disk is the root file system containing base folders and programs which are needed to run the kernel. In order to make the RAM disk, you should follow the process described below. Even though you can make the RAM disk image by using commands that will be described in the following paragraph, it is provided in the EOS-uClinux kernel source directory. Therefore, you can use it without having to create one.

Although any user can run most of the user programs such as “busybox” and “init”, only root can make the RAM disk, which produces files with the “ext2” file system and uses the “mount” command, which requires root privileges.

Here is a process to make the RAM disk.

- Create an empty file with the size of 2M bytes. In this example the file is called “ramdisk-eos.img”.
  ```
  > dd if=/dev/zero of=ramdisk-eos.img bs=1k count=2048
  ```
- Convert it into the “ext2” file system.
  ```
  > mke2fs ramdisk-eos.img
  ramdisk-eos.img is not a block special device. Proceed anyway? (y,n) ; type ‘y’ when it asks to process.
  ```
- Create a directory and then mount it.
  ```
  > mkdir eos-ramdisk
  > mount –t ext2 –o loop ramdisk-eos.img eos-ramdisk
  ```
- Make a base directory for Linux, or a directory that you want to add inside the “eos-ramdisk” folder.
  ```
  > cd eos-ramdisk
  > mkdir bin dev etc home lib mnt proc root sbin usr
  ```
- Change modes of all application programs executable and then copy them into the “bin” or “usr/bin” directories.
  ```
  For example, you could do the following:
  > chmod +x busybox.elf.bflt
  > cp busybox.elf.bflt eos-ramdisk/bin/busybox
  ```
- Set a device to the “eos-ramdisk/dev” folder. Major and minor numbers of each device are defined in the kernel or each device driver of users. You must give the numbers defined in the “rd-eos.img.gz” file for each device.
  ```
  > cd eos-ramdisk/dev
  ```
> mknod ttyS0 c 204 16
> mknod mtdblock0 b 31 0
> ...

- Edit the configuration file for the Linux in the “eos-ramdisk/etc” folder.
- Do any additional jobs that you want.
- Unmount the “eos-ramdisk” and then compress it.
  > unmount eos-ramdisk
  > gzip –vf9 ramdisk-eos.img
- The above process will produce “ramdisk-eos.img.gz.”
4 How to run uClinux on the EOS demo Board (Nand Flash Boot Method)

The EOS can execute a program by copying 1KB of data at address 0 of the Nand Flash to the internal SRAM of the EOS based on the settings of external jumpers. The initial program is limited to 1 Kbytes in size. The program contains base hardware information and the copy command from Nand Flash to SDRAM and then a program control jump to the SDRAM.

Figure 4-1 shows details of external pins JP14 to JP21 that can be used by the Nand boot. The values of these pins are automatically mapped into the CFGR register at address 0x1FF00004 when the board is on. When the computer is reset, these values are not changed. Figure 5-1 depicts the mapping relationship between contents in the CFGR and the values of the external pins JP14 to JP21.

![Figure 4-1: The mapping between the CFGR and board jumper.](image)

If the pin JP18 (CFGR: 4) is set to 1, then it activates an execution mode, which reads and copies the content of the 1K-byte size of the Nand to the SRAM. The width of the bus can be set either 8 or 16 bits depending on the value of the pin JP20. In order to run the EOS board with the auto boot mode, you must set the pin JP20 to 1, which selects 8 bits for the bus width, since it is set to 8 bits for the default bus width in the Nand in the EOS board. The pin JP17 determines how many cycles are required for writing an address in the Nand. You must select 3 cycles for the auto boot mode.

The JP14, JP15, JP16 pins are for the users. You can operate them for your own purpose by assigning bit0 to bit2 in the CFGR with the values that you want.
By default, the Nand boot code selects an appropriate boot mode either the uClinux auto boot or the bootloader modes by reading values of JP14, JP15, and JP16 pins. Figure 4-2 through 4-4 demonstrate settings of the jumpers in each mode. Figure 5-4 shows settings for the normal boot ROM mode.

Figure 4-2: Auto boot mode (uClinux boot.)

Figure 4-3: Auto boot mode (bootlader boot.)

Figure 4-4: Normal boot mode (boot from rom).

In order to boot your system using the Nand Flash, you should ensure each program is stored in the proper location in the Nand flash. Figure 4-5 illustrates the default address mapping defined in the Nand boot code.
You can write the Nand boot codes and bootloader by using the JTAG. **In order to write in the Nand using JTAG, you must set the jumpers to the normal boot mode.** Here are some examples of how to use the JTAG:

- Nand boot code: `eos_jtag -v -nand -f nand_boot.bin`
- Bootloader code: `eos_jtag -v -nand -base 0x2000 -f eos_loader_ram.bin`

You should keep in mind to use the “-base” option for the address in the flash. Without the “-base” option, it writes from address 0 in the flash. For more information, read the JTAG users’ manual.

However, it takes long time to write the kernel and RAM disk images using the JTAG, therefore, we recommend you to run the bootloader program with the auto boot mode and then to load the file to the Nand flash by using “tftp”. See the bootloader users’ manual to learn more about the “tftp”.

**Figure 4-5:  Nand Flash Address Map.**
5 User Defined Setting

You can save some information, such as an IP address, in the MTD Nand device driver when power is off. The “init” program in the Ram disk automatically executes commands defined in the “/etc/rc” and “/mnt/flash0/rc2” files. You may add instructions in the “/mnt/flash0/rc2” which are executed during boot. Code 6-1 shows “/mnt/flash0/rc2” including the network setup. You can modify it to add the settings for the IP address for your computer using any text editor such as “vi”.

5.1 IP Address

```
ifconfig eth0 210.102.3.120 netmask 255.255.255.128
route add default gw 210.102.3.10
cp resolv.conf /etc
cat /etc/motd
```

Code 5-1: The configuration file, “/mnt/flash0/rc2”, that you can modify.

- Setting up an IP address for your computer: you can add it after “eth0” in the “/mnt/flash/rc2” file.
- Setting up a name server for your computer: you must edit the “/mnt/flash/resolv.conf.” It should be placed in the “/etc”.

Figure 5-1 is contents of the “rc2” and “resolv.conf” in the flash.

![Example of rc2 and resolv.conf](image)

Figure 5-1: Examples of the “rc2” and “resolv.conf” stored in the Nand flash.

5.2 Ethernet MAC Address

You can save MAC address by using ‘flashparam’ command in bootloader with ‘mac0’ or ‘mac1’ option.
Figure 5-2: View mac address stored in the nand flash

Figure 5-3: Setting mac address
6 How to use the Bootloader

6.1 Introduction

In general, an embedded system uses a bootloader, not a general purpose BIOS as is used in a PC.

6.2 Functionality of the Bootloader

- Initializing hardware
  Initialize a CPU clock, Memory timing, an interrupt, and an UART.
- Booting Linux.
  It stores a kernel image to the SDRAM and then jumps the program control to an address for the kernel image.
- Downloading the kernel image and the file system.
  In the following subsections, we will focus on this.

6.3 Instruction Sets

- File Download
  - Serial (sdownload)
  - Ethernet (tftp, bootp)
- Flash write, erase
- Kernel booting

![Figure 6-1: List of bootloader command](image)

6.4 Setting up “bootp” and “tftp” for a Host PC (a Linux Server)

To confirm whether “bootp” and “tftp” are installed, use the following commands:

```bash
> rpm -qa | grep bootp
> rpm -qa | grep tftp
```
If they are not installed, you can install the following files from the RPM directory in the CD:

```bash
bootp-2.4.3-7.i386.rpm
tftp-server-0.17-9.i386.rpm
```

```
> rpm -ivh bootp-2.4.3-7.i386.rpm tftp-server-0.17-9.i386.rpm
```

The following steps describe how to configure bootp and tftp after installation. A “bootp” and a “tftp” require a server and a client. The host plays a server role while the target board is a client. An IP address is required in order to communicate using “bootp”. “bootp” is a protocol that allows an IP address to be acquired while booting. Even though a “dhcp” does the same function as “bootp”, “bootp” is commonly used for target boards without any OS since the size of “dhcp” is bigger and the logic is more complicated.

“bootp” announces the MAC address of its own Network Interface Card (NIC) to a server when a system is booted, and acquires the IP address according to the MAC address. Let’s define a server for the bootp. The bootp and the tftp are under control of a “xinetd.” As the Red hat Linux uses the “xinetd”, we are going to do so. Of course, if you want to use an “inetd”, you can install and use it.

The “xinetd” manipulates them while it creates a file for each daemon in the “/etc/xinetd.d/.” It is also possible to be manipulated by the “/etc/xinetd.conf.” We recommend using the first method since it is easier to see and manipulate to use the file. You must move to the “/etc/xinetd.d” directory and create a configuration file first.

```
# vi bootps

service bootps
{
    disable = no
    socket_type = dgram
    protocol = udp
    wait = no
    user = root
    server = /usr/sbin/bootpd
}
```

The above file is used by “xinetd” to manipulate a boot server. Let’s create a table containing addresses including the IP for a bootp client.

```
# vi /etc/bootptab

EOS : \ :ht=1 \ :ha=0x009907090A0A \n```

18
You have done configuring the bootp server. Now you need to configure a “tftp.” To do so, you only need to modify a little as there exists a configuration file for the “tftp” already.

```bash
# vi /etc/xinetd.d/tftp
```

```
service tftp
{
    socket_type = dgram
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot  // a location to store an image at the host
    disable = no
}
```

If you have done configuration, you need to run the “xinetd” again as following:

```bash
# /etc/rc.d/init.d/xinetd restart
# netstat -nau
```

```
upd    0      0 0.0.0.0:67           0.0.0.0:*  
upd    0      0 0.0.0.0:69           0.0.0.0:*  
```

By typing a “netstat,” with a “nau” option, the computer tell you that port #67 and #69 are opened. The port #67 is for the “bootps”, while the port #69 is for the “tftp.” If you want to see name of each daemon, instead of the port number, you should type in “netstat” with the option of “au”, not “nau.” The following is an example of what you might see:

```bash
# netstat -au
```

```
upd    0 0 0.0.0.0:bootps 0.0.0.0:*  
upd    0 0 0.0.0.0:tftp 0.0.0.0:*  
```

Finally, the location to be downloaded is the “/tftpboot” which you set in the “/etc/xinetd.d/tftp.” If you store the
kernel or the RAM disk to the “/tftpboot”, then you are now ready to download in the target.

```
[root@SoftLinux /tftpboot]# pwd
/tftpboot
[root@SoftLinux /tftpboot]# ls
linux.bin  rd-eos.img.gz
```

6.5 How to Use Bootp, tftp

① File download

There are two ways to download current EOS images to a target machine: “sdownload” and “tftp.” We recommend using the “tftp” which is using an Ethernet as it is faster than the “sdownload” that is using serial port.

We will present how to use the “tftp” only since you can find how to use the “sdownload” from the manual accompanied.

② bootp

Figure 6-2 shows a screen in which the host receives the IP address. From now on, you can use the “tftp.”

```
aeloader> bootp
```

![Figure 6-2: An example of the bootp.](image)

③ tftp

It downloads files in the host to the SDRAM using the “tftp” protocol.

How to use: `tftp <file> {kernel, ramdisk}`

```
aeloader> tftp linux.bin kernel
It downloads the “linux.bin” file from the host to the target address, kernel of the SDRAM.
aeloader> tftp rd-eos.img.gz ramdisk
```
It downloads the rd-eos.img.gz file from the host to the target address, ramdisk of the SDRAM.

Figure 6-3: An example of the tftp.

6.6 Serial Downloading

You also can download the kernel and the RAM disk images using serial such as the Graphical user Interface (GUI) for “bootloader” and “sdownloader”. Even though it is slower than a method using an Ethernet, it is useful when there is no Ethernet ready.

- How to use:
  - Type the following to get ready for receiving data to your host:

```bash
ealoader> download
```

- Run the serial download program and connect inside a window of your host.

CPU type : 32bit
Select Port : COM1
Select Baudrate : 57600
- After connection is made, finds out the address of the image that should be downloaded and downloads.
- If downloading is accomplished, you will see the following message from the bootloader of a target machine.

```
aeloader> download
Command OK!
Successfully Received data!
```

- You must select 'Cancel' button for the Serial Download.
- The above process is to download one file. If you want to get another file transferred, you have to repeat it from the beginning.

```
An example:
```aeloader> download  // downloading the kernel
Command OK!
Successfully Received data!
```
```
aeloader> download  // downloading the RAM disk
Command OK!
Successfully Received data!
```

6.7 Writing and Erasing the Nand Flash

① Writing flash

What writing a flash does is downloading images to a Nand flash region that are downloaded from a SDRAM using a
“tftp” and a “bootp.” You do it because the kernel and the file system images downloaded to the SDRAM will be erased after a power of a computer is off.

**How to use: flash {kernel, ramdisk}**

```bash
aeloader> flash kernel
   ➔ Writing the kernel image downloaded to the SDRAM to the kernel address in the flash.
```

```bash
aeloader> flash ramdisk
   ➔ Writing the RAM disk image downloaded to the SDRAM to the RAM disk address in the flash.
```

The above instructions can only be executed after the proper image is downloaded to the SDRAM using ‘tftp’. After downloading the images, you can run the kernel as following:

```bash
aeloader> boot
```

![Figure 6-5: An example of transferring the serial data from the bootloader.](image)

**② flash erase**

This instruction removes contents in a Nand flash. You can delete any region followed by a variable defined as following:

**Usage:** erase {ramdisk, kernel, user0, user1}

**Usage:** eraseall

```bash
aeloader> erase ramdisk   ➔ removing the RAM disk
aeloader> erase kernel    ➔ removing the kernel
aeloader> erase user0     ➔ removing the user0
aeloader> erase user1     ➔ removing the user1
```

Refer Figure 4-5 illustrating the Nand flash address map in the Nand flash.
If you want to remove all regions of the flash, run the following command:

```
aeloader> eraseall
```
Part III. How to use application programs

7 Busybox Utility

It is same the way to use “ls”, “mkdir”, and “clear” utilities in the “busybox” as that to do them in the general Linux. We will explain some of them in which you may be interested.

7.1 Ping

- Usage: ping HOST [PORT]
- Figure 7-1 shows an example of using the “ping”:

![Figure 7-1: An example of using the ‘ping’ program.](image1)

7.2 Telnet

- Usage: telnet HOST [PORT]
- Figure 7-2 illustrates the output of running telnet.

![Figure 7-2: An example of the telnet program.](image2)
7.3 Tftp

You must register the IP and MAC addresses in the Linux server by referring to the bootloader manual.

- To download a file: tftp –g –r [a filename in a server] [a server address]
- To send a file: tftp –p –l [local filename] [server address]
- Figure 7-3 is an example of using the ‘tftp’ program

![Figure 7-3](image1)

Figure 7-3: An example of using the ‘tftp’ program.

7.4 Wget

“wget” is a program for downloading via the FTP or Http protocols.

- How to use: wget –P [a name of a folder downloaded] [URL]
- Figure 7-4 is an example of using the “wget” program:

![Figure 7-4](image2)

Figure 7-4: An example of using the ‘wget’ program.
An Example for a Keypad and a device driver Keypad

The code below is an example of a keypad. The way to use a keypad drive is to use the “read” function after opening the “dev/eoskeypad” device. If a key is entered, it creates an interrupt and stores it into the internal buffer (32 bytes in size) which is defined for EOS_KEYPAD_BUF_SIZE in “driver/char/eos_keypad.c.” The read function for a keypad reads one letter from the buffer at a time. If there is a key value, it returns 1. Otherwise, it returns 0. The “eos_getkey()” function waits until a key is entered inside the ‘while’ statement.

```c
#define EOS_KEYPAD_DEV "/dev/eoskeypad"
unsigned char eos_getkey(int dev);
#define KEY_BUF_SIZE 128

int main(void)
{
    int dev;
    int nread;
    int i;
    unsigned char key = 0;

    printf("EOS Keypad Test \n\n");

    if ((dev = open(EOS_KEYPAD_DEV, O_RDONLY)) < 0) {
        printf("Open Error %s\n", EOS_KEYPAD_DEV);
        return EXIT_FAILURE;
    }
    for (;;) {
        key = eos_getkey(dev);
        if (key == 'X')
            break;
        printf("Key : %c\n", key);
    }
    close(dev);
    return 0;
}

unsigned char eos_getkey (int dev)
{
    unsigned char key;
    while (!read(dev, &key, 1));
    return key;
}
```

Code 8-1: An example program for the EOS Keypad.

Figure 8-1 shows a definition of a keypad driver given.
If you change a definition of a key map, you must update it in the eos_keypad.c file as follows:

```c
Static unsigned char eos_decode_keypad[64] =
{
    0x00, '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-','+','-', '/','\', '!', '@', '#', '$', '%', '
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
    'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

You can update a value of a key code map inside an application program by changing a value of the “ioctl” function by using the SET_KEYCODE instruction shown in the above example program. However, it is slow because it takes time to copy a key code array. We recommend you to use the method to change a key map inside the kernel if possible.

## 9 Example of RTC

A device driver for the Dallas DS1302 RTC is provided with the EOS uClinux kernel distributed after 2004-02-02. The DS1302 can be controlled by GPIO. The driver is designed to connect the RST pin of the DS1302 to PIO12, the SCLK pin to PIO14, the I/O pin to PIO13, respectively.

The DS1302 device driver can count up to 2099 from 2000. There are two RTC ioctl command: RTC_RD_TIME which reads the RTC, and RTC_SET_TIME which changes a value of the RTC. Figure 9-1 illustrates a process of reading the current time, printing it out, changing a value of the RTC to 2004/12/24 23:50:10, and then showing the result.
Figure 9-1: Example of the RTC.
10  NFS (Network File System)

NFS makes it possible to use a directory in a remote computer as if it was located in your local computer. When you want to load a program into a target board, it is painful to copy it in the RAM disk and then load it by “tftp”, or to transfer a file by “ftp.” By using NFS, you can get rid of the painful process of making a RAM disk. If you use NFS, you can use directories stored in a server from the “/eos-nfs” directory on a client board (EOS demo board), as they are resident in the local board.

10.1 Configuring the kernel

In order to use a NSF for a server/client, you must configure some options inside the kernel.

- NFS server

The kernel should be compiled with NFS functionality. By default, most of the distributions are already compiled with NFS support. If it is not compiled with NFS functionality, recompile the kernel with the following options:

File systems ➔

Network File System ➔

[*] NFS file system support
[*] Provide NFSv3 client support
[*] NFS server support

Answer "No" to the following option:

Root file system on NFS

Compile the kernel after answering all questions then install the new kernel.

- NFS Client:

The kernel provided in the CD should have NFS installed. You can confirm the following options.

File systems ➔

Network File System ➔

[*] NFS file system support
[*] Provide NFSv3 client support

10.2 Setting up the NFS (Server Linux)

- Packages required

nfs-utils-[version].i386.rpm : [version] is for the package.

- Contains tools related to daemons for the NFS server.
- “showmount”: checks information about a remote host of the NFS.
- portmap-[version].i386.rpm

First check whether the package for NFS and portmapper are installed by typing the following:

```
# rpm –qa | grep  nfs
nfs-utils-0.3.1-13hl  : shows that it is installed already.
# rpm –qa | grep portmap
portmap-4.0-38hl
```

If they are not already installed, install them by running the following commands:

```
# rpm –ivh nfs-utils-[version].i386.rpm
# rpm –ivh portmap-[version].i386.rpm
```

### 10.3 Running portmapper (Server Linux)

Run portmapper by typing the following.

```
# /etc/rc.d/init.d/portmap start
```

→ Confirm the mapping information (portmapper uses port 111.)

```
# rpcinfo –p
```

<table>
<thead>
<tr>
<th>Program</th>
<th>Version</th>
<th>Type</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>2</td>
<td>tcp</td>
<td>111</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>udp</td>
<td>111</td>
</tr>
</tbody>
</table>

Configure so that it is automatically executed during the boot sequence.

```
# chkconfig --level 35 portmap on
# chkconfig --level 35 nfs on
```

### 10.4 Setting up a NFS server (setting up a host)

- Create a distribution directory that you want to share as follows.

  ```
  # mkdir /eos_nfs_host
  Edit the /etc/exports file to add "/eos_nfs_host 210.102.3.0/255.255.255.0(rw,no_root_squash)".
  # vi /etc/exports
  /eos_nfs_host 210.102.3.0/255.255.255.0(rw,no_root_squash)
  ```

- Export it by typing the following.

  ```
  # exportfs –r
  ```
• Confirm whether it is exported properly.

```bash
# cat /var/lib/nfs/etab
/eos_nfs_host  210.102.3.0/255.255.255.0(rw,async,wdelay,hide,secure,no_root_squash,
no_all_squash, subtree_check,secure_locks,mapping=identity,anonuid=-2,anongid=-2)
```

• Start the NFS daemon.

```bash
#/etc/rc.d/init.d/nfs start    : portmapper should be running already.
```

• Start the NFS lock manager.

```bash
#/etc/rc.d/init.d/nfslock start
Starting NFS file locking services:
Starting NFS statd: [Check]
(Normally, this is not run manually, but automatically run by the kernel.)
```

• Check the exported mount information as follows.

```bash
#/etc/rc.d/init.d/nfslock start
#
```

• Finally, check the daemon by typing the following.

```bash
#/etc/rc.d/init.d/nfslock start
#
```

```bash
Program Version Type Port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100005 1 udp 1025 mountd
100005 1 tcp 1025 mountd
100005 2 udp 1025 mountd
100005 2 tcp 1025 mountd
100005 3 udp 1025 mountd
100005 3 tcp 1025 mountd
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100021 1 udp 1026 nlockmgr
100021 3 udp 1026 nlockmgr
100021 4 udp 1026 nlockmgr
```

32
10.5 Setting up a NFS client (Setting up a target board)

- “portmapper” should be running prior to executing the following process.
- Mount the directory exported by the server.
  
  ```
  # mkdir /eos_nfs : NFS mount directory name
  # portmap &
  # mount -t nfs 210.102.3.97:/eos_nfs_host /eos_nfs
  ```
- Check whether it is mounted properly by running the following command.
  
  ```
  # mount
  ```

10.6 Mount NFS using the shell script

- Since '/mnt/flash0' directory is mounted to the 'mtdblock', it is not removed even after rebooting.
  
  ```
  # vi /mnt/flash0/nfs.sh
  #!/bin/sh
  if [ ! -d "/eos_nfs" ]; then
    mkdir /eos_nfs
    portmap &
    mount -t nfs 210.102.3.97:/eos_nfs_host /eos_nfs
  fi
  
  # chmod 755 /mnt/flash0/nfs.sh
  ```

Run the script that you have written after a reboot.

  ```
  # /mnt/flash0/nfs.sh
  ```

10.7 NFS Test

- Server
  
  ```
  # cd /eos_nfs_host
  # vi test
  EOS uCLinux Test!!!
  ```

- Client
  
  ```
  # cd /eos_nfs
  # vi test
  ```