SUZAKU Software Manual

Version 1.1.0

January 31, 2005

Atmark Techno, Inc.
http://www.atmark-techno.com/

Table of Contents

List of Tables

List of Figures

List of Examples

1. Introduction

1.1. About This Manual

This manual provides information on the following areas necessary for using the MicroBlaze™ software processor and uClinux on the SUZAKU.

- Basic operations
- Overwriting the Flash memory
- Building a Kernel and Userland
- Customizing
- Application Software Development

Unless otherwise specified, the work PC is assumed to run a Linux-based OS.

We hope the information in this manual will help you to get the best functionality out of the SUZAKU.

1.2. About the Fonts

In this manual fonts are used according to the following conventions.

Table 1-1 Fonts

Fonts	Description
Fonts in text	Text
[PC ~]$ **ls**	Prompt and user input character strings

1.3. Conventions in Command Input Examples

The command input examples contained in this manual are based on the assumed execution environment associated with the respective display prompt. The directory part "/" will differ depending on the current directory. The home directory of each user is represented by "~".

Table 1-2 Relationship between Display Prompt and Execution Environment

Prompt	Command Execution Environment
[PC /]#	To be executed by a privileged user on the Work PC
[PC /]$	To be executed by a general user on the Work PC.
[SUZAKU /]#	To be executed by a privileged user on the SUZAKU.
[SUZAKU /]$	To be executed by a general user on the SUZAKU.

1.4. Precautions

1.4.1. Safety Precautions

Before using the SUZAKU, read the following safety precautions carefully to assure correct use.

> This product uses semiconductor components designed for generic electronics equipment such as office automation equipment, communications equipment, measurement equipment and machine tools. Do not incorporate this product into devices such as medical equipment, traffic control systems, combustion control systems, safety equipment, etc. which can directly threaten human life or pose a hazard to the body or property due to malfunction or failure. Moreover, products incorporating semiconductor components can be caused to malfunction or fail due to foreign noise or a surge. To ensure there will be no risk to life, the body or property even in the event of malfunction or failure, be sure to take all possible measures in the safety system design, such as using protection circuits like limit switches or fuse breakers, or system multiplexing.

1.4.2. Handling Notes

To avoid degradation, damage, malfunction, or fire, the following safety precautions must be observed when operating the product.

- Input Voltage
  Do not attempt to apply an input voltage higher than 3.3V+5%. Use caution in polarity.

- Interface
  Do not connect signals other than specified to each interface (external, I/O, RS232C, Ethernet or JTAG).
  Use caution in polarity.
  Be careful with the input/output direction of signals.

- Modification
  Do not make any modifications other than adding external I/O connectors or JTAG connectors (CON2, CON3, CON4, CON5 and CON7).

- FPGA Programming
  Be careful not to program the FPGA in a way that may cause a collision of a peripheral circuit (including on-board components) and a signal (i.e. output of the same signal from two devices).
  Use caution when programming the FPGA.

- Power-on
  While the board and peripheral circuits are being supplied power, be sure not to connect or disconnect FPGA I/O as well as JTAG connectors.

- Static Electricity
  This board incorporates CMOS devices. Before using the board, store it safely in the provided antistatic package.

- Latch-up
  Due to excessive noise or a surge from the power supply or input/out, or rapid voltage fluctuations, the CMOS devices incorporated in the board can cause a latch-up. Once a latch-up occurs, this situation continues until the power supply is disconnected. As a result, it can damage devices. It is recommended to take safety measures such as adding a protection circuit into the noise-susceptible input/output lines or not sharing a power supply with devices that can be the cause of noise.

- Impact/Vibration
  Do not apply a high impact such as a drop or collision.
  Do not put this product on anything vibrating and/or rotating. Do not apply strong vibration or centrifugal force.

- Excessive High/Low Temperatures and High Humidity
  Do not use the product under environments which subject it to excessively high/low temperatures or high humidity.

- Dust
  Do not use the product in dusty areas.

1.4.3. Precautions before Using Software

The software and documentation contained in this product is provided “AS IS” without warranty of any kind including any warranty of merchantability or fitness for a particular purpose, reliability, correctness or accuracy. Furthermore, Atmark Techno, Inc. does not guarantee any outcomes resulting from the use of this product.

1.5. Acknowledgements

The software used in the SUZAKU consists of Free Software and Open Source Software. Free Software and Open Source Software are the achievements of many developers from around the world. We would like to take this opportunity to thank all these developers.

uClinux is supported by the achievements of D. Jeff Dionne, Greg Ungere, David McCulloughu and all people participating in the uClinux development list.

uClibc and Busybox have been developed and are maintained by Eric Andersen.

The uClinux original port, which runs on the MicroBlaze processor architecture, is the achievement of John Williams (Embedded System Research Group, University of Queensland, Brisbane, Australia).

2. Overview of the SUZAKU

The SUZAKU is a generic small-form FPGA board. This software manual describes uClinux, which runs on the 32-bit soft processor MicroBlaze that is incorporated in the SUZAKU at shipment.

2.1. MicroBlaze

The MicroBlaze is a Xilinx 32-bit soft processor based on the Harvard architecture. It is a basic RISC CPU and has the following features: 32bit instruction word length, 32 general purpose registers, IBM's On-chip Peripheral Bus and Fast Simplex Link support and a hardware multiplexer.

For more information, refer to the Xilinx Company Web Site.

2.2. uClinux

uClinux[1] is a Kernel derived from Linux. By eliminating the dependency on a MMU, Linux can run on CPUs that do not have a MMU. This project started when D. Jeff Dionne ported [2] Linux to the Motorola 68000, which does not have a MMU. It is now being developed at uClinux.org.

Other than the Kernel, uClinux.org distributes a Tool Chain and Userland applications called uClinux-dist. This manual assumes the use of uClinux-dist.

The MicroBlaze uClinux project is implemented by John Williams and deals with porting uClinux to MicroBlaze and distributing the Linux Tool Chain.

[1] As of development versions 2.5, the developments of uClinux were also merged into the main Linux kernel.
[2] D. Jeff Dionne: "Linux port to straight 68000 (no MMU)" http://lkml.org/lkml/1998/2/12/31

3. Before Getting Started

3.1. Preparations

Please prepare the following before using the SUZAKU.

- Work PC
  One PC that can run Linux and provides one serial port is required for software development.

- D-Sub9-pin Cross Cable
  A D-Sub9-pin (female-to-female) cable for cross connections.

- D-Sub9-pin to 10-pin Conversion Cable
  The D-Sub9-pin to 10-pin conversion cable is required to connect the D-Sub9-pin connector to the pin header (10-pin) on the board.

- Supplied Development Kit CD-ROM (hereafter called the "Supplied CD-ROM")
  This CD-ROM contains various manuals and source code related to the SUZAKU.

- Serial Communication Software
  Minicom communication software is used in this manual.

- DC3.3V Power Supply
  Prepare a power supply that has a DC3.3V output.

4. Getting Started

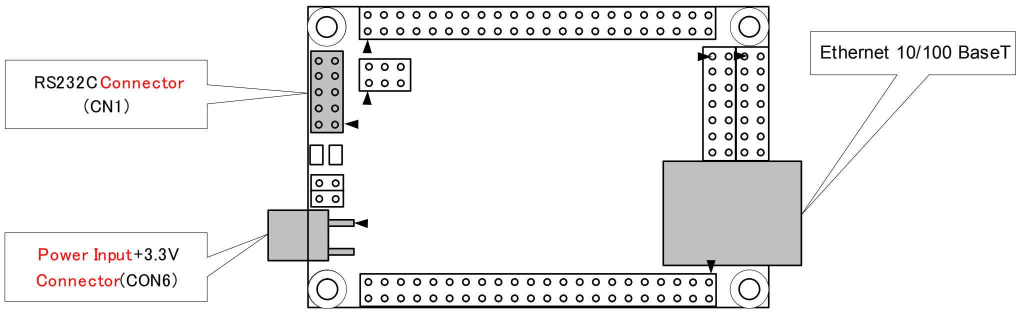
This chapter shows how to connect the various cables to the factory set SUZAKU and then get it running. The SUZAKU comes with the soft processor MicroBlaze along with MicroBlaze uClinux stored in the Flash memory. The SUZAKU with factory default settings is set up to obtain an IP address using DHCP. Therefore, please prepare a DHCP server network environment.

The following sections provide information starting from how to connect cables to the SUZAKU to accessing it via the Web.

4.1. Connecting Cables

Prepare one power cable, D-Sub9-pin to 10-pin cable and LAN cable and connect them to the appropriate connectors according to Figure 4-1 SUZAKU Connector Layout Diagram (Figure 4-1). The SUZAKU itself does not have a power switch. Make sure to keep the power supply switched off until all preparations are complete.

Figure 4-1 SUZAKU Connector Layout Diagram



Caution:

Connect a D-Sub 9-pin to 10-pin conversion cable to the RS232C connector (CON1) with the white triangular mark on the connector to be fitted to that on the SUZAKU board. Be careful not to connect the cable in an incorrect orientation. The equipment may be damaged.

4.2. Serial Communication Software

The SUZAKU uses a serial port for console connection. To read the information output from the SUZAKU console or send information to the SUZAKU console, serial communication software is required. In this manual we use the GPL-provided Minicom communication software. The supplied CD-ROM that comes with the board contains Debian and Redhat packages and source code. For more information, refer to the “tools” directory contained in the supplied CD-ROM. If Minicom is not installed, install it according to the installation procedure described in Installing Minicom” in Section 7.4.

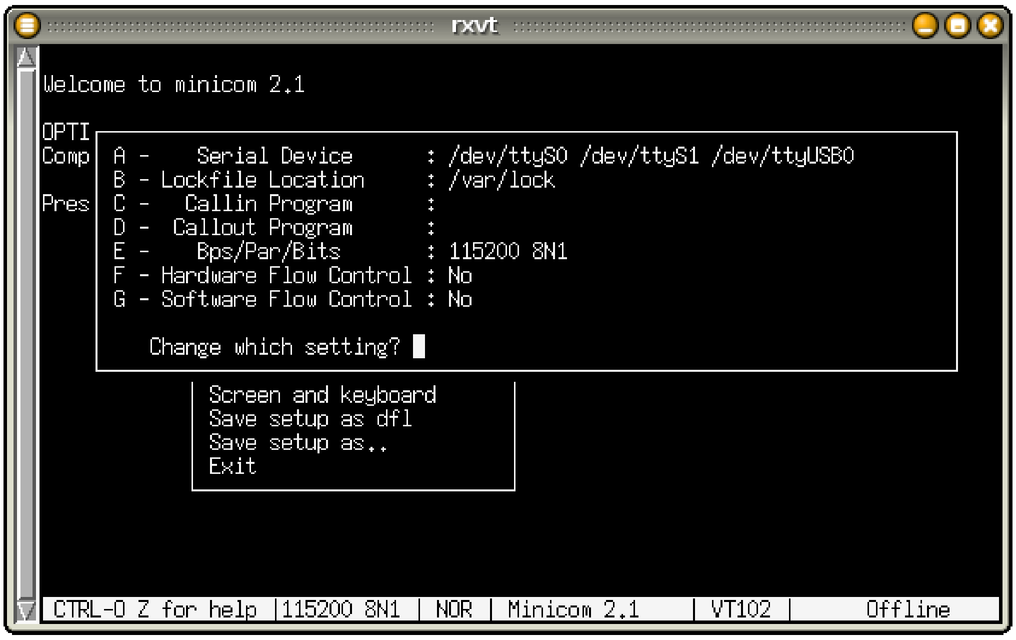
When the Minicom installation is complete, then set the Minicom parameters. Parameter settings for serial communication with the SUZAKU are shown in Table 4-1.

Table 4-1 Serial Communication Parameter Settings

Parameter	Values
Data Transfer Rate	115,200bps
Data Length	8-bit
Stop Bit	1-bit
Parity Check	None
Flow Control	None

Run Minicom by adding the "-s" option to it. The option "–s" will automatically take you to the Minicom configuration screen. Set the serial port parameters according to "Table 4-1 Serial Communication Parameter Settings".

Figure 4-2 Minicom Settings



```
Welcome to minicom 2.1

OPTI
Comp  A -    Serial Device      : /dev/ttyS0 /dev/ttyS1 /dev/ttyUSB0
      B - Lockfile Location     : /var/lock
Pres  C -   Callin Program      :
      D -  Callout Program      :
      E -    Bps/Par/Bits       : 115200 8N1
      F - Hardware Flow Control : No
      G - Software Flow Control : No

         Change which setting?

              Screen and keyboard
              Save setup as dfl
              Save setup as..
              Exit

 CTRL-O Z for help |115200 8N1 | NOR | Minicom 2.1    | VT102 |      Offline
```

Save the changes and exit, and then reactivate Minicom.

TIPS

It appears that in many default Minicom setups modem initialization is performed during start-up. You can omit the modem initialization command by deleting the AT command for initialization in setup or by adding the "–o" option to Minicom to start-up.
If you do not have read and write privilege to the serial port, Minicom will fail to start-up. Be sure to check the privileges of the serial port. For more information, refer to the Minicom manual or your OS manual.

Example 4-1 Starting minicom

```
[PC ~]$ minicom –o

Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Nov 12 2003, 19:21:57.

Press CTRL-A Z for help on special keys
```

4.3. Power-on

Make sure that the boot mode selection jumper is set to open. The factory default SUZAKU has multiple boot modes. The boot mode selection jumper is used to select these boot modes. For more information, refer to "Boot Mode" on Section 6.2. Before the SUZAKU is turned on, make sure that Minicom is activated.

After the SUZAKU is turned on, the boot log as shown in Example 4-2 appears in the Minicom screen. If it does not appear, refer to Troubleshooting in Section 13.

Example 4-2 uClinux Boot Log

```
Switching to 2nd stage bootloader...
Copying kernel...done.
Linux version 2.4.27-uc0-suzaku0 (atmark@build) (gcc version 2.95.3-4 Xilinx EDK
6.1 Build EDK_G.11) #1 Sat Jan 29 16:23:31 JST 2005
On node 0 totalpages: 4096
zone(0): 4096 pages.
zone(1): 0 pages.
zone(2): 0 pages.
CPU: MICROBLAZE
Console: xmbserial on UARTLite
Kernel command line:
Calibrating delay loop... 25.44 BogoMIPS
Memory: 16MB = 16MB total
Memory: 13336KB available (1037K code, 1824K data, 44K init)
Dentry cache hash table entries: 2048 (order: 2, 16384 bytes)
Inode cache hash table entries: 1024 (order: 1, 8192 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 4096 (order: 2, 16384 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Microblaze UARTlite serial driver version 1.00
ttyS0 at 0xffff2000 (irq = 1) is a Microblaze UARTlite
Starting kswapd
xgpio #0 at 0xFFFF5000 mapped to 0xFFFF5000
Xilinx GPIO registered
SMSC LAN91C111 Driver (v2.0), (Linux Kernel 2.4 + Support for Odd Byte) 09/24/01
 -     by Pramod Bhardwaj (pramod.bhardwaj@smsc.com)
eth0: SMC91C11xFD(rev:1) at 0xffe00300 IRQ:2 MEMSIZE:8192b NOWAIT:0 ADDR:
00:11:0c:00:15:55
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
Suzaku MTD mappings:
  Flash 0x800000 at 0xff000000
flash: Found an alias at 0x400000 for the chip at 0x0
 Amd/Fujitsu Extended Query Table v1.1 at 0x0040
flash: Swapping erase regions for broken CFI table.
number of CFI chips: 1
cfi_cmdset_0002: Disabling fast programming due to code brokenness.
Creating 7 MTD partitions on "flash":
0x00000000-0x00400000 : "Flash/All"
0x00000000-0x00080000 : "Flash/FPGA"
0x00080000-0x000a0000 : "Flash/Bootloader"
```

```
0x003f0000-0x00400000 : "Flash/Config"
0x000a0000-0x003f0000 : "Flash/Image"
0x000a0000-0x00210000 : "Flash/Kernel"
0x00210000-0x003f0000 : "Flash/User"
FLASH partition type: auto 4MiB
uclinux[mtd]: RAM probe address=0x8013cb88 size=0x18f000
uclinux[mtd]: root filesystem index=7
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 1024)
VFS: Mounted root (romfs filesystem) readonly.
Freeing init memory: 44K
Mounting proc:
Mounting var:
Populating /var:
Running local start scripts.
Mounting /etc/config:
Populating /etc/config:
flatfsd: Created 4 configuration files (150 bytes)
Setting hostname:
Setting up interface lo:
Starting DHCP client:
Starting inetd:
Starting thttpd:
eth0:PHY remote fault detected

SUZAKU login:
```

4.4. Logging-in

Enter user name “root” in the SUZAKU login prompt to login. The default password is “root”.

Table 4-2 User Name and Password in SUZAKU Default Setting

User Name	Password
root	root

4.5. Command Input

As an example of command input, the following example shows how to display network settings using an ifconfig command. The default SUZAKU network configuration uses dhcp.

Example 4-3 Command Input with ifconfig

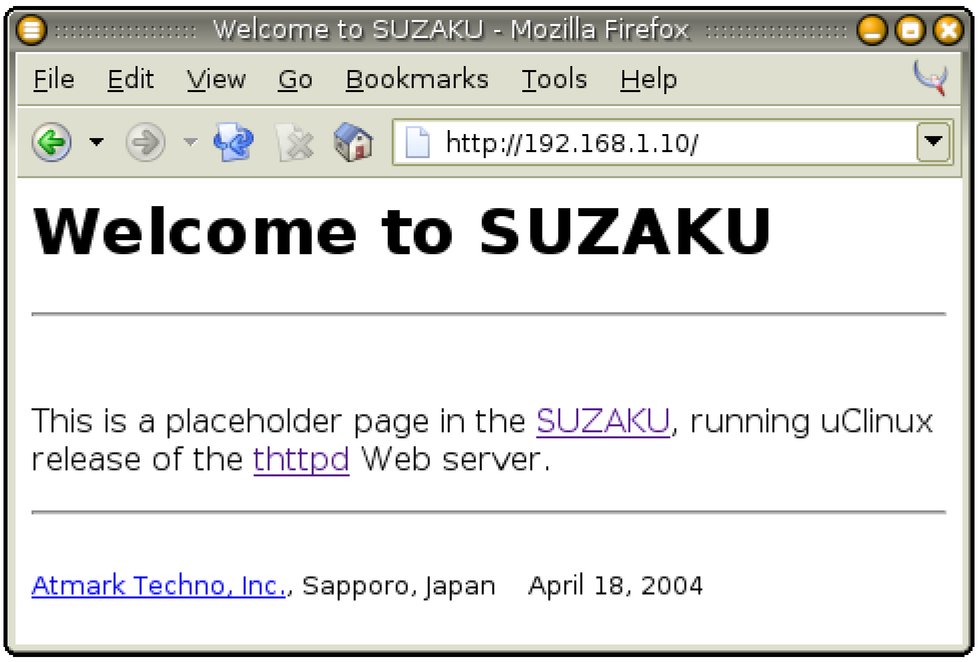
```
[SUZAKU ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:11:0C:00:10:00
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:347 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          Interrupt:2 Base address:0x300

[SUZAKU ~]#
```

4.6. Web Site

With SUZAKU's default settings, a small HTTP server called thttpd is set to run. By accessing the IP address (192.168.1.10 in the example) as shown in Example 4-3 Command Input with ifconfig using your web browser, you can check the SUZAKU is running.

Figure 4-3 SUZAKU Web Page


Welcome to SUZAKU - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.10/

Welcome to SUZAKU

This is a placeholder page in the SUZAKU, running uClinux release of the thttpd Web server.

Atmark Techno, Inc., Sapporo, Japan April 18, 2004

4.7. Shutting the SUZAKU down

The SUZAKU takes measures to prevent damage to the root file system from abrupt power cuts by defining it as read-only. Therefore, you can exit the SUZAKU simply by turning it off. However, the Flat Filesystem which the SUZAKU uses to save configuration files cannot handle power cuts that occur while information is being written. Therefore, power cuts may result in the loss of data on the Flat Filesystem.

For more information, refer to the Flat Filesystem or the Flat Filesystem Daemon manual.

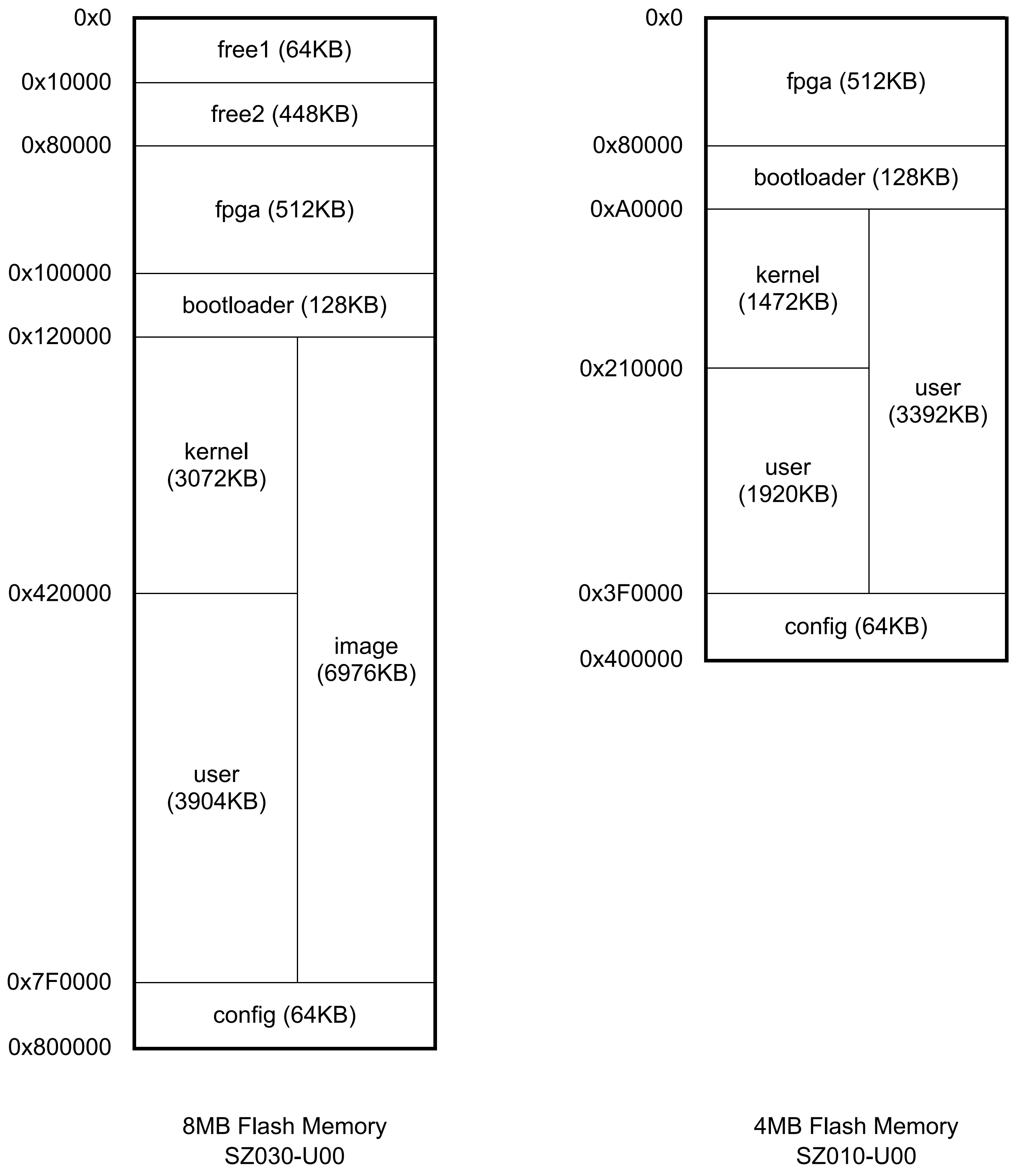
Caution

It is possible to customize the SUZAKU to change the root file system from the factory default of ROMFS to JFFS2. In these circumstances, cutting power to the SUZAKU may result in the loss of data that would have normally been saved. For more information, refer to the JFFS2 manual.

5. Flash Memory Map

The SUZAKU Flash memory is divided into several areas called regions as shown in Table 5-1 Flash Memory Map to facilitate the software utilization on SUZAKU.

Table 5-1 Flash Memory Map



Warning

The fpga region contains the FPGA configuration data. Writing improper data to this area can cause heat generation, degradation or damage to the SUZAKU or peripheral equipment due to the malfunction of the SUZAKU. To restore the SUZAKU to a normally functioning state, it will be necessary to reprogram the FPGA configuration data. For more information, refer to the Hardware Manual.

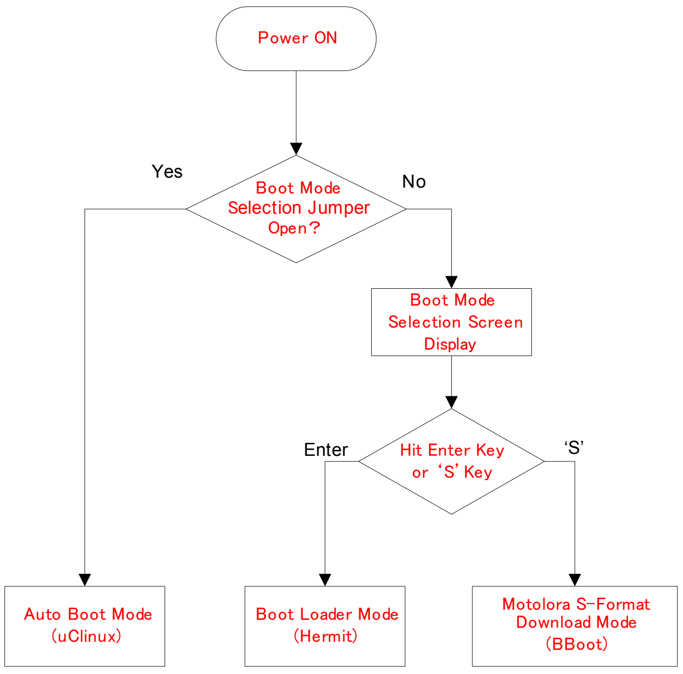
6. Boot Flow and Mode

The SUZAKU with default settings has three boot modes. Before uClinux is activated on the SUZAKU, two boot loaders are running. This chapter provides a simple description of the boot flow. It then provides information on the three boot modes available on the SUZAKU.

6.1. Boot Flow

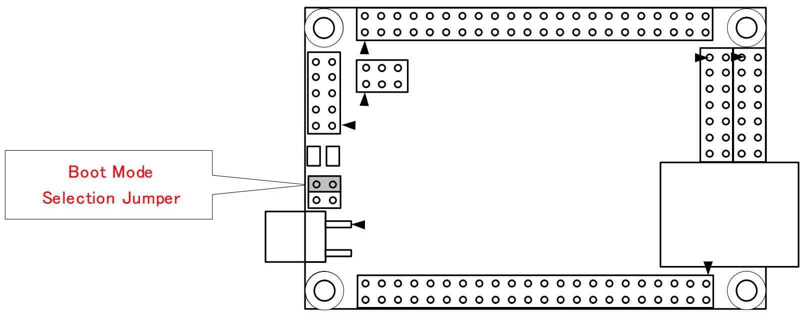
The boot flow is shown in Figure 6-1 Flow of Boot Mode.

Figure 6-1 Flow of Boot Mode


Power ON
Yes
Boot Mode Selection Jumper Open?
No
Boot Mode Selection Screen Display
Enter
Hit Enter Key or 'S' Key
'S'
Auto Boot Mode (uClinux)
Boot Loader Mode (Hermit)
Motolora S-Format Download Mode (BBoot)

The boot mode flow is determined by the boot mode jumper on the SUZAKU. For the location of the boot mode jumper, refer to Figure 6-2 Boot Mode Jumper. When this jumper is set to "open", uClinux is started in auto boot mode. While, if this jumper is set to "short", the SUZAKU will move to the boot mode selection screen.

Figure 6-2 Boot Mode Jumper



The boot mode jumper on the SUZAKU is set to "open" or "auto boot mode" when shipped.

6.2. Boot Mode

There are three boot modes available on the SUZAKU. The following sections describe each mode.

6.2.1. Auto Boot Mode

In this mode, uClinux is automatically booted up when the SUZAKU is activated.

6.2.2. Boot Loader Mode

This mode is only used when the Boot Loader is to be activated. The SUZAKU ships with a high performance boot loader and down loader called Hermit in the Flash memory. In this mode, Hermit waits for user input. By entering commands to Hermit, you can read the information contained in the memory or write a uClinux image to the Flash memory.

To go into the boot loader mode, set the boot mode jumper to "short" and then hit the Enter key as shown in Example 6-1 Boot Mode Selection Screen. When SUZAKU's default boot loader Hermit is activated, a screen as shown in Example 6-2 Activating Hermit appears.

Example 6-1 Boot Mode Selection Screen

```
Welcome to BBoot - SUZAKU's first stage bootloader and S-Record downloader

hit Enter key to activate second stage bootloader or
hit 'S' key to download S-Record
```

Example 6-2 Activating Hermit

```
Hermit v1.3-armadillo-7 compiled at 22:22:42, Apr 26 2004
hermit>
```

To write data to the Flash memory using Hermit, you must first activate SUZAKU in this mode. For more information, refer to 12.2 "Overwriting Flash Memory Using Hermit.

6.2.3. Motolora S-Format Download Mode

This mode is used to overwrite the boot loader in the Flash memory if it malfunctions for some reason. To go into the mode, you can set the boot mode selection jumper to “short”, and after the Example 6-1 Boot Mode Selection Screen appears, enter “S” from the keyboard.

A Motolora S-format downloader called BBoot is incorporated in the SUZAKU at shipment. This BBoot is stored in the area called Block RAM within the FPGA, not in the Flash memory. Therefore, even if uClinux or Hermit are mistakenly damaged, it can be used as long as the MicroBlaze maintains a normal operation. Since BBoot is the first program that runs when the SUZAKU is activated, it is referred to as the “first stage boot loader”.

For more information on how to write in Motolora S-format using BBoot, refer to 12.3 "Overwriting Flash Memory in Motolora S-Format.

7. Preparations of Development Environment

This chapter shows how to prepare a development environment for MicroBlaze uClinux. Preparations include uncompressing the tool chain to /usr/local/microblaze-elf-tools/, setting the PATH environment variable and preparing source code.

7.1. Installing the Tool Chain

The tool chain is stored under the “cross-dev” directory in the supplied CD-ROM. It can also be downloaded from the home page of the MicroBlaze uClinux project.

The tool chain is a tar archive file compressed using gzip. Restore it under the /usr/local/microblaze-elf-tools/ directory.

The following shows the series of commands when installing it from the supplied CD-ROM.

```
[PC ~]$ su -
[PC ~]# mkdir -p /usr/local/microblaze-elf-tools/
[PC ~]# cd /usr/local/microblaze-elf-tools/
[PC microblaze-elf-tools]# mount /cdrom
[PC microblaze-elf-tools]# tar -xvzf /cdrom/cross-dev/microblaze-elf-tools-20040315.tar.gz
[PC microblaze-elf-tools]# ls
bin   include   info   lib   man   microblaze   share
[PC microblaze-elf-tools]# exit
[PC ~]$
```

7.2. Setting Environment Variable

The directory that contains the executable file of the tool chain is added to the PATH environment variable to make it easier to use. Since the method differs dependent on the type of shell, refer to the manual relevant to the shell you are using.

The following shows an example for “bash”.

```
[PC ~]$ export PATH=$PATH:/usr/local/microblaze-elf-tools/bin
[PC ~]$
```

7.3. Preparing Source Code

uClinux.org provides source code for the kernel, library and applications as a single package named the ”Full Source Distribution”. The file name of this Full Source Distribution provided by uClinux.org is uClinux-dist-YYYYMMDD.tar.gz. Another version of the uClinux-dist file which contains some modifications has been is created for the SUZAKU. A uClinux-dist file for the SUZAKU is contained in the “source” directory of the CDROM with the file name “uClinux-dist.tar.gz”. For ease in tracing daily updates of source code, the uClinux-dist file for the SUZAKU contains CVS information issued by uClinux.org.

uClinux-dist.tar.gz can be restored to any location where it will be easy to use. In this manual it is restored under "~/".

```
[PC ~]$ su -
[PC ~]# mount /cdrom
[PC ~]# exit
[PC ~]$ tar -xvzf /cdrom/source/uClinux-dist.tar.gz
[PC ~]$ ls
uClinux-dist
[PC ~]$
```

7.4. Installing Minicom

The following are examples of the installation procedure under Debian and Redhat.
For more information, refer to your OS manual.

Example 7-1 Installing Minicom (using dpkg under Debian)

```
[PC /cdrom/tools]# dpkg -i minicom_2.1-4.woody.1_i386.deb
[PC /cdrom/tools]#
```

Example 7-2 Installing Minicom (using rpm under Redhat)

```
[PC /cdrom/tools]# rpm -i minicom_2.1-1-rh7.3.i386.rpm
[PC /cdrom/tools]#
```

7.5. Installing Hermit

Install Hermit on the work PC. Hermit consists of two programs, one for the target side that runs on the SUZAKU and the other for the host side that runs on the work PC and communicates with the program on the SUZAKU. The Hermit to be installed on the target side also plays an important role as the boot loader used to activate uClinux. While the Hermit for the host side is used to overwrite the Flash memory on the SUZAKU.

The supplied CDROM contains a Hermit package under the "tools" directory. Install it according to your OS. The following are examples of the installation procedure under Debian and Redhat.

Example 7-3 Installing Hermit (Debian)

```
[PC /cdrom/tools]# dpkg -i hermit_1.3-armadillo-7_i386.deb
[PC /cdrom/tools]#
```

Example 7-4 Installing Hermit (Redhat)

```
[PC /cdrom/tools]# rpm -i hermit-1.3_armadillo-7.i386.rpm
[PC /cdrom/tools]#
```

8. Creating a uClinux Image

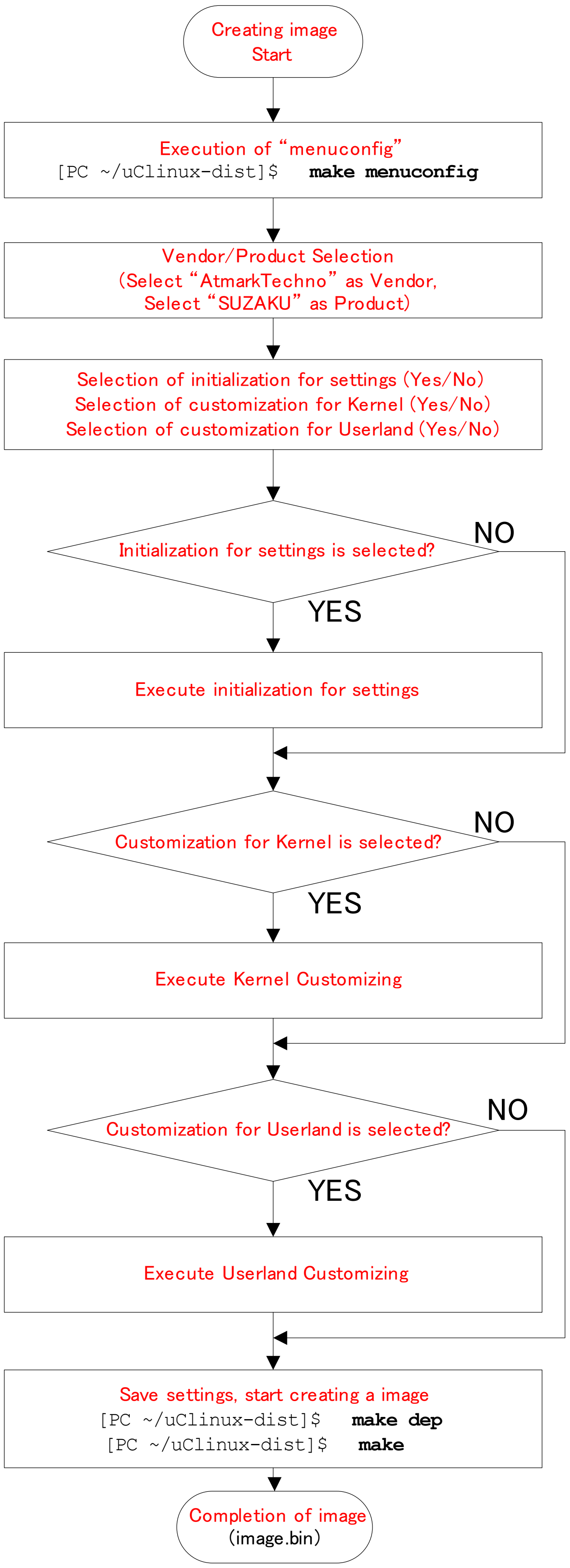
This chapter shows how to create an image file that is used when writing to the Flash memory using the tool chain and source code previously installed. uClinux-dist enables customization in the same way as the build system of the Linux Kernel. It will be easy to handle for people who compile the Linux Kernel by themselves. This manual shows an example using menuconfig.

First, this example creates the default image file that is written to the Flash memory of the SUZAKU when it ships.

8.1. Work Flow

The work flow to create an image file is illustrated in Figure 8-1 Work Flow to Create an Image File.

Figure 8-1 Work Flow to Create an Image File



Creating image Start
Execution of "menuconfig"
Vendor/Product Selection (Select "AtmarkTechno" as Vendor, Select "SUZAKU" as Product)
Selection of initialization for settings (Yes/No)
Selection of customization for Kernel (Yes/No)
Selection of customization for Userland (Yes/No)
Initialization for settings is selected?
NO
YES
Execute initialization for settings
Customization for Kernel is selected?
NO
YES
Execute Kernel Customizing
Customization for Userland is selected?
NO
YES
Execute Userland Customizing
Save settings, start creating a image
Completion of image (image.bin)

```
[PC ~/uClinux-dist]$   make menuconfig
```

```
[PC ~/uClinux-dist]$   make dep
[PC ~/uClinux-dist]$   make
```

The procedure is divided into three categories: Initialization, Kernel and Userland.

Caution

Selecting Initialization initializes both Kernel and Userland. Furthermore, all existing work done before selecting Initialization will be lost.

8.2. Menuconfig

The uClinux-dist build system allows the use of config, menuconfig and xconfig as with the Linux Kernel. This manual shows how to build uClinux-dist using menuconfig.

First go to the uClinux-dist directory which was restored in 7.3."Preparing Source Code" and then enter the "**make** menuconfig" command.

Example 8-1 Executing make menuconfig

```
[PC ~]$ cd uClinux-dist
[PC ~/uClinux-dist]$ make menuconfig
```

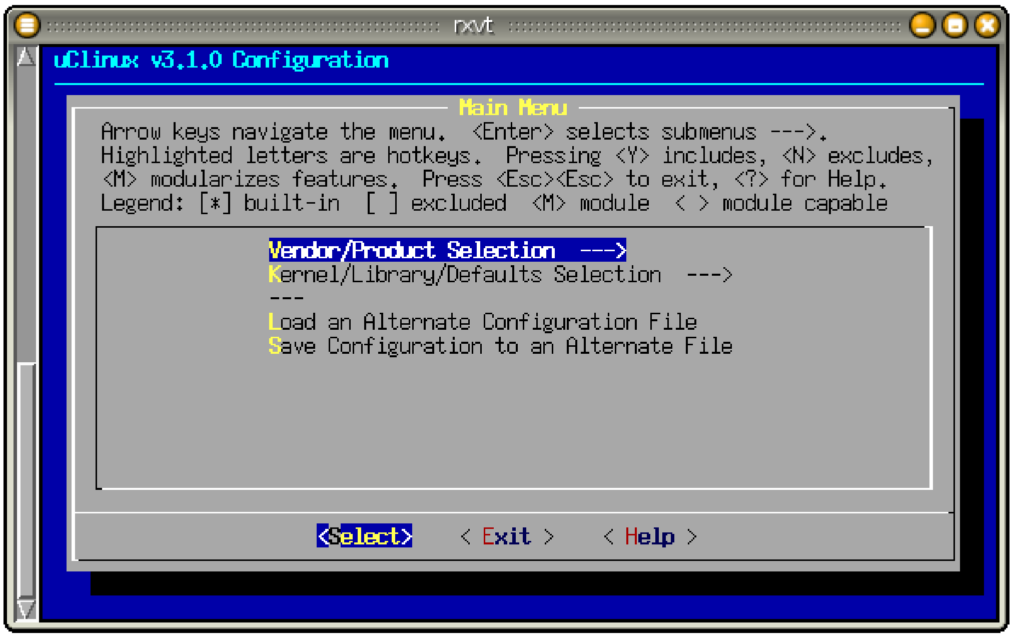
Caution

At the first execution the "make menuconfig" command compiles a screen control program that needs the ncurses library. This ncurses library must therefore be preinstalled. Dependent on type of OS, you will need to install not only the ncurses library package but also the development package for that library.

8.2.1. Main Menu

After the command is entered, a screen as shown in Figure 8-2 uClinux v3.1.0 Configuration Main Menu" appears. This is the uClinux-dist main menu.

Figure 8-2 uClinux v3.1.0 Configuration Main Menu



```
rxvt
uClinux v3.1.0 Configuration
                         Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

              Vendor/Product Selection  --->
              Kernel/Library/Defaults Selection  --->
              ---
              Load an Alternate Configuration File
              Save Configuration to an Alternate File

                  <Select>    < Exit >    < Help >
```

The menuconfig allows movement to a specific selection menu by using the arrow keys on the keyboard. The symbol **--->** at the right side of a selection menu represents that you can move to a submenu. Make sure that the **<select>** button at the bottom of the screen is highlighted when selecting a menu and then hit the Enter key.

This screen allows movement to the Vendor/Product selection menu and the Kernel, Library and Default selection menu.

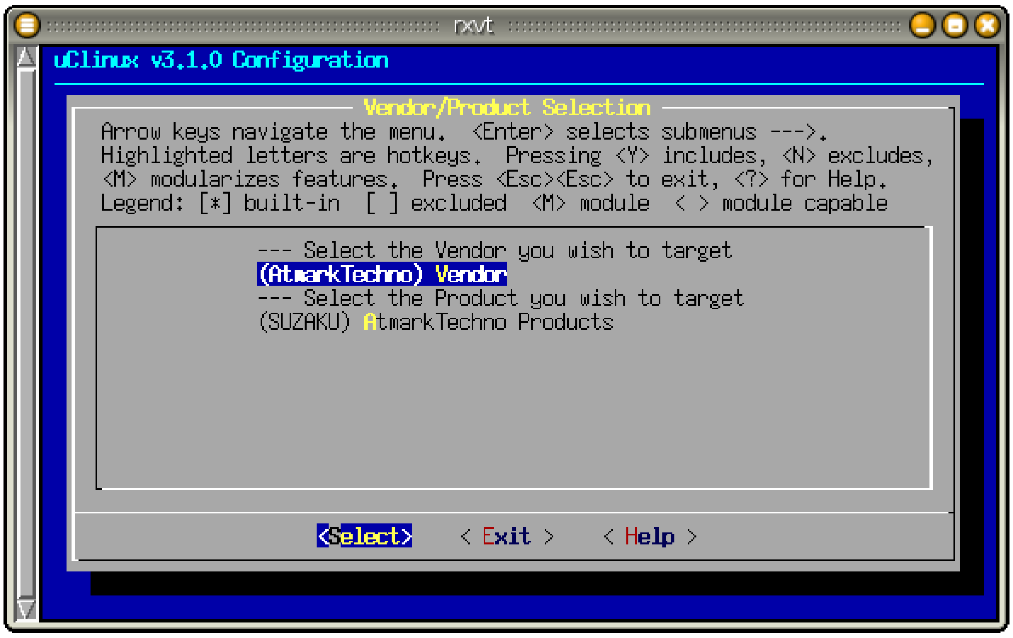
8.2.2. Vendor/Product Menu

Select Vendor/Product from the main menu and move to the submenu screen.

By default SnapGear is selected. You can select other vendors by putting the cursor on SnapGear (Vendor) and hitting the Enter key. Select AtmarkTechno from the list.

Selecting AtmarkTechno as shown in Figure 8-3 Selecting AtmarkTechno enables you to select the SUZAKU by Product.

Figure 8-3 Selecting AtmarkTechno



```
rxvt
uClinux v3.1.0 Configuration
                    Vendor/Product Selection
 Arrow keys navigate the menu.  <Enter> selects submenus --->.
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

              --- Select the Vendor you wish to target
              (AtmarkTechno) Vendor
              --- Select the Product you wish to target
              (SUZAKU) AtmarkTechno Products

              <Select>    < Exit >    < Help >
```

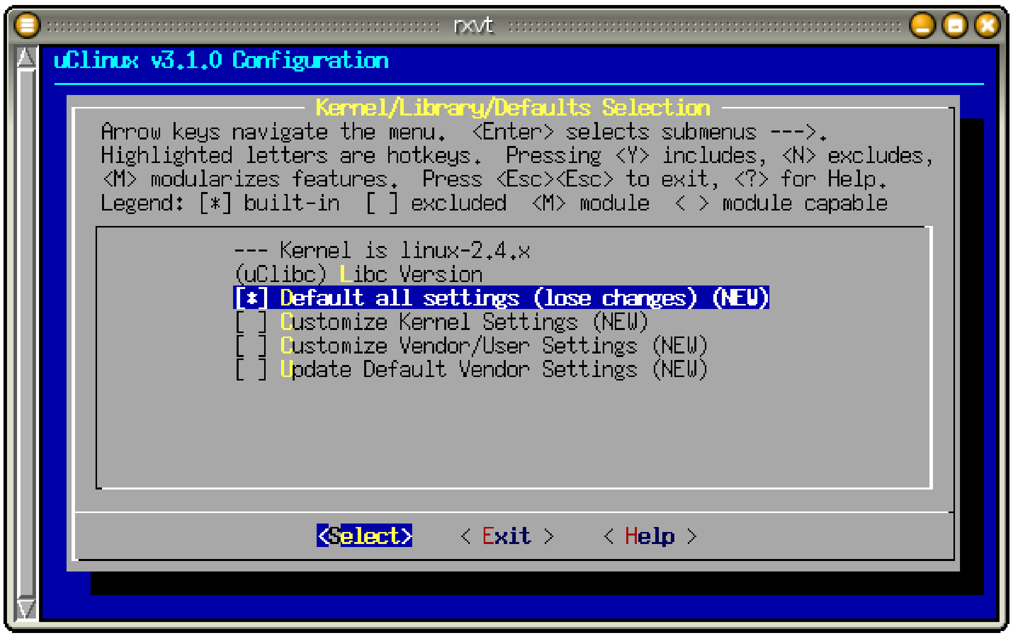
Select Exit to return to the main menu.

8.2.3. Kernel/Library/Defaults Menu

Select Kernel/Library/Defaults Selection from the main menu to go to the submenu.

The symbol [ ] at the left side of the menu represents the selection state. This time, please select **Default all settings (lose changes) (NEW)**. Move the cursor using the up and down arrow keys and make a selection using the space bar. When selected, an asterisk (*) is shown in [ ].

Figure 8-4 Kernel/Library/Defaults Selection



```
uClinux v3.1.0 Configuration
                 Kernel/Library/Defaults Selection
 Arrow keys navigate the menu.  <Enter> selects submenus --->.
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

              --- Kernel is linux-2.4.x
              (uClibc) Libc Version
              [*] Default all settings (lose changes) (NEW)
              [ ] Customize Kernel Settings (NEW)
              [ ] Customize Vendor/User Settings (NEW)
              [ ] Update Default Vendor Settings (NEW)

                  <Select>    < Exit >    < Help >
```

After selecting Default all settings (lose changes), select Exit to return to the Main Menu. If you select Exit in Main Menu, you will be asked whether or not to save the settings. Select **Yes** to save them.

After a configuration log appears on the screen, you will return to the prompt.

8.3. Build

Building is the name given to actually compiling or creating image files. Building is done using a make command. Input at the prompt the following shown in Example 8-2 Build Command.

Example 8-2 Build Command

```
[PC ~/uClinux-dist]$ make dep; make
```

The first make dep is a command used to resolve the dependency of the Linux Kernel. It will be a familiar command for users with experience of compiling Linux Kernels up to version 2.4.

The next make command performs the entire build processes and finally creates an image file that can be written to the Flash memory. If the build process is successfully completed, an image.bin file will be created under the uClinux-dist/images/ directory.

For information on how to write an actual image file to the SUZAKU, refer to Section 12 “OverwritingOverwriting Flash Memory”.

[3] In this manual the image file refers to a binary file combining the uClinux Kernel and Userland. Writing this binary file to the Flash memory using Hermit will make uClinux bootable on the SUZAKU.

9. uClinux Configuration

This chapter shows how to customize a uClinux image using mainly the uClinux-dist menu.

9.1. Customizing Kernel Settings

To customize the Kernel configuration, select Customize Kernel Settings from the Kernel/Library/Defaults Selection menu in uClinux Configuration. When you select this menu and quit out from the main menu of uClinux v3.1.0 Configuration, the Linux Kernel Configuration menu screen appears automatically. Linux Kernel Configuration itself is the same as the main version of Linux.

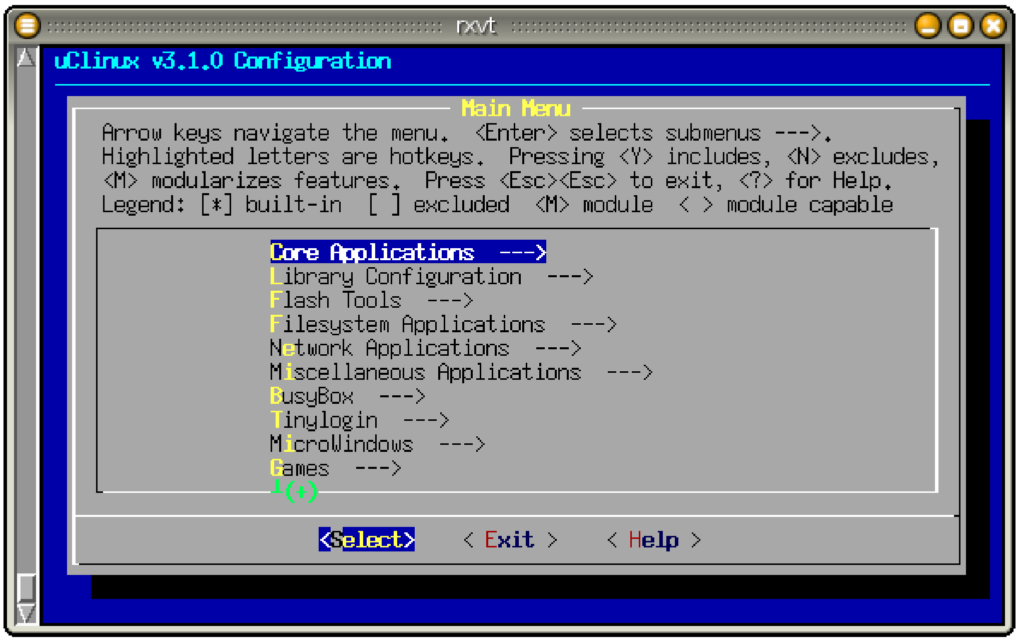
When you complete the customization of Kernel settings, select **Exit**. When asked whether or not to save the settings, select **Yes**.

If you don't need to customize Userland, execute a Build. For information on "build", refer to Section 8.3 "Build".

9.2. Customizing Userland Settings

To customize Userland, select Customize Vendor/UserSettings from the Kernel/Library/Defaults Selection menu of uClinux Configuration. When you select this menu and terminate the uClinux v3.1.0 Configuration main menu, the Userland Main Menu appears.

Figure 9-1 Userland Main Menu



```
uClinux v3.1.0 Configuration
                        Main Menu
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

              Core Applications  --->
              Library Configuration  --->
              Flash Tools  --->
              Filesystem Applications  --->
              Network Applications  --->
              Miscellaneous Applications  --->
              BusyBox  --->
              Tinylogin  --->
              MicroWindows  --->
              Games  --->
              ↓(+)

          <Select>    < Exit >    < Help >
```

In the Userland customization screen, you have the options of whether or not to add applications provided by the uClinux Full Source Distribution to the created image or to customize the applications to be added. Furthermore, you can change the root user password and so on.

uClinux-dist has a collection of more than 150 applications. In the uClinux Configuration menu they are classified into several categories.

- Core Application

  Core Application contains the basic applications necessary for running as a system. This section allows selection of “init” for system initialization and “login” for user authentication.

- Library Configuration

  Library Configuration allows selection of libraries required by applications.

- Flash Tools

  Flash Tools allow selection of applications associated with Flash memory. In the SUZAKU a network update application called netflash is selected in this category.

- Filesystem Applications

  Filesystem Applications allow selection of applications associated with file system. In the SUZAKU flatfsd is selected. Others included are mount, fdisk, ext2 file system, reiser file system and Samba.

- Network Applications

  Network Applications allow selection of applications associated with networking. In addition to dhcpcd-new, ftpd, ifconfig, inetd and thttpd used in the SUZAKU, ppp and a wireless network utility are also included.

- Miscellaneous Applications

  Miscellaneous Applications include other applications not belonging to the above categories. Generic Unix-based commands (cp, ls, rm, etc.), editors, audio-related programs, and a script language interpreter are also included.

- Busybox

  Busybox can be customized here. Busybox is a single command having multiple command functions and has a good track record in embedded Linux systems. Since Busybox provides many customization functions, it is classified into a separate section.

- Tinylogin

  The Tinylogin application also provides multiple command functions associated with authentication such as login, passwd and getty. Since it provides many customization functions, it is classified into a separate section.

- MicroWindows

  MicroWindows is a graphical window environment targeted towards embedded equipment. It is well suited to the development of LCD-based equipment.

- Game

  This is a game. No further explanation needed.

- Miscellaneous Configuration

  Numerous configuration options are contained here. The SUZAKU root user password can also be changed here.

- Debug Builds

  Contains numerous debug options. This category is used when debugging an application during development.

There is no guarantee that every application will run on all architectures. However, in most cases, they can be run with minor modifications (at source code level or with Makefile).

Many developers hold discussions on modifications via the uClinux.org mailing list. If interested in these discussions, please feel free to participate in them.

9.3. Creating ZFLAT-based Kernel

This section shows how to create a ZFLAT-based Kernel. ZFLAT refers to an execution file in compressed FlatBinary Format. For information about the Flat Binary Format, refer to 11. “Flat Binary Format”.

Execute a “make menuconfig” commnad in the uClinux-dist directory and then select Customize Kernel Settings from the Kernel/Library/Defaults Selection menu. Select Exit from Main Menu. When you are asked whether or not to change settings, select Yes.

Go to General setup from the Linux Kernel Configuration main menu. Select “Enable ZFLAT support” beneath “Kernel support for flat binaries”. Select Exit from the Main Menu and save the settings to quit.

Execute “make clean && make dep && make” command to generate image.bin. The generated image Kernel allows execution of a ZFLAT file.

For information on how to compress a Flat Binary Format execution file, refer to 11.2 Compressing Executable Files.

10. Adding Your Own Applications to the Image

This chapter explains how to create Hello World and run it on the SUZAKU. Compiling is performed outside of the uClinux-dist package. This is referred to as an Out of Tree Compile (OTC). Sample code is contained in the "sample" directory of the supplied CDROM.

Even if compiling is performed outside of uClinux-dist, since it uses the uClinux-dist supplied libraries and Makefile, a uClinux-dist directory previously built for the SUZAKU is required. If not yet built, refer to Section 8 " Creating a uClinux Image" to build it.

10.1. Creating a Directory

You can create a directory used for the work anywhere. In this manual we use "~/hello".

Example 10-1 Preparing Hello World Directory

```
[PC ~]$ mkdir hello
[PC ~]$
```

10.2. Creating a Makefile

The supplied CDROM contains Makefile sample templates in "sample/hello/Makefile". First copy this file to the work directory you created.

Example 10-2 Hello World Makefile

```
ifndef ROOTDIR
ROOTDIR=../uClinux-dist  ........................................ ①
endif
ROMFSDIR   = $(ROOTDIR)/romfs
ROMFSINST = romfs-inst.sh
PATH        := $(PATH):$(ROOTDIR)/tools

UCLINUX_BUILD_USER = 1  ......................................... ②
include $(ROOTDIR)/.config
LIBCDIR = $(CONFIG_LIBCDIR)
include $(ROOTDIR)/config.arch


EXEC = hello  ................................................... ③
OBJS = hello.o  ................................................. ④


all: $(EXEC)

$(EXEC): $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
        -rm -f $(EXEC) *.elf *.gdb *.o

romfs:
        $(ROMFSINST) /bin/$(EXEC)

%.o: %.c
        $(CC) -c $(CFLAGS) -o $@ $<  ............................ ⑤
```

① If ROOTDIR is not specified, it is assumed that the uClinux-dist directory exists in parallel to the directory you are currently in. If the uClinux-dist directory does not exist there, specify it with ROOTDIR.
② Define UCLINUX_BUILD_USER. Defining this variable automatically selects the Userland compiler option.
③ Enter the executable file name to be generated in the variable EXEC. In this manual it is set to “hello”.
④ Specify the object file to be used for the above executable file. If you want to specify multiple object files, separate each object file with a space.
⑤ This is a pattern rule to convert the C source code into the same name of object code. For more information, refer to the Make manual.

10.3. C Source Code

The C source code of Hello World is shown in Example 10-3 Hello World Source Code.

Example 10-3 Hello World Source Code

```
#include <stdio.h>

int main(int argc, char * argv[])
{
        printf("Hello SUZAKU World!¥n");
        return 0;
}
```

This is the standard Hello World which can be seen in any C textbook. Save it as “hello.c”. It is compiled into hello.o according to Makefile.

10.4. Compiling

Do a “make” as shown in the following example. If “make” is successful, three files will be generated: hello.o, hello.gdb and hello. hello is the executable file.

Example 10-4 Execution of “make”

```
[PC ~/hello]$ make
[PC ~/hello]$ ls
Makefile   hello   hello.c   hello.gdb   hello.o
[PC ~/hello]$
```

10.5. Installing into ROMFS Directory

The user area of the image file is generated from the uClinux-dist/romfs directory. Therefore, to include your own application in the image to be written to Flash memory, the application must be copied to the “romfs” directory.

If, like this time, there is only one generated executable file, the copying is easy. However, it is not so easy when generating several executable files and copying the configuration and data files. To cope with this, the Makefile template provides a romfs target.

Example 10-5 Installing Hello World into ROMFS Directory

```
[PC ~/hello]$ make romfs
romfs-inst.sh /bin/hello
[PC ~/hello]$ ls ../uClinux-dist/romfs/bin/hello
../uClinux-dist/romfs/bin/hello
[PC ~/hello]$
```

10.6. Creating and Executing an Image File

Lastly, move to the uClinux-dist directory and create an image file.

By specifying the image target with the “make” command, after the Userland image file (romfs.img) based on the romfs directory is created, it is combined with the Kernel image file (linux.bin) to form the one image file (image.bin) that is to be written into the Flash memory.

Example 10-6 Creating an Image File

```
[PC ~/hello]$ cd ../uClinux-dist
[PC ~/uClinux-dist]$ make image
        :
        :
        :
[PC ~/uClinux-dist]$ ls images
image.bin   linux.bin   romfs.img
[PC ~/uClinux-dist]$
```

Download the created image file to the SUZAKU and execute the “hello” file. For information on how to download the image file, refer to 12 “Overwriting Flash Memory.

Example 10-7 Execution

```
[SUZAKU ~]# hello
Hello SUZAKU World!
[SUZAKU ~]#
```

11. Flat Binary Format

This chapter describes the Flat Binary Format that is one of uClinux's features. The size of the executable binary files represents a critical problem in the embedded systems targeted by uClinux. The ELF used by generic Linux provides a format rich in flexibility but the size is too large. So, most uClinux adopt a new binary format similar to the traditional "a.out" format.

This section first describes the features of theFlat Binary Format and then provides information on how to compress executable files and change the stack size.

11.1. Flat Binary Format Features

The Flat Binary Format has the following features.

- Simplicity
  This simple format design contributes to the execution speed and size of the binary file. Although it has less flexibility than ELF, it can be said that it is a necessary tradeoff for embedded systems.

- Compression

  The Flat Binary Format is a compressible format. There are two compression types, whole file compression and data area only compression. The executable file will be uncompressed when it is loaded, so the activation speed is slow compared to a non-compressed executable file. Since there is no difference between it and a non-compressed file if once it is activated, the format is suitable for programs such as a residential processes which do not repeatedly activate and shutdown.

- Stack Size Field
  The Flat Binary Format has a stack size field that can be changed without re-compiling. In CPUs without a MMU, since it is difficult to extend the stack area dynamically, it has a stack area with fixed stack size. This field can be changed with the tool called "flthdr". Furthermore, it is also possible to specify its stack size at compiling.

- XIP (eXecute In Place)
  The Flat Binary Format is also compatible with XIP. XIP is an abbreviation for "eXecute In Place" (spot execution), and generally refers to the ability of executable files to run directly on ROM where they are stored, without being copied to RAM.

11.2. Compressing Executable Files

This example shows how to compress the Hello World program which was created in the previous chapter. The later part also shows how to specify compression when compiling.

Caution
With the default Kernel, you cannot execute a compressed Flat Binary Format (ZFLAT) file. To create a ZFLAT-based Kernel, refer to 9.3 Creating ZFLAT-based Kernel.

11.2.1. Compressing a Compiled Binary File

Using "flthdr" we compress a compiled binary file. The "flthdr" is a program that is used to edit or display Flat Binary Format files. It is contained in the MicroBlaze tool chain under the "mb-flthdr" name, and is hereafter referred to as "mb-flthdr".

The following example represents an executable file created in a normal compiling.

Example 11-1 Normal Flat Binary Format

```
[PC ~/hello]$ make
[PC ~/hello]$ mb-flthdr hello
hello
    Magic:          bFLT
    Rev:            4
    Entry:          0x50
    Data Start:     0x3e60
    Data End:       0x4bf0
    BSS End:        0x6bf0
    Stack Size:     0x1000
    Reloc Start:    0x4bf0
    Reloc Count:    0x53
    Flags:          0x1 ( Load-to-Ram )
[PC ~/hello]$
```

Next, compile it using “mb-flthdr”.

Example 11-2 Compressed Flat binary Format

```
[PC ~/hello]$ mb-flthdr -z hello ---------------------------------------------------        ①
zflat hello --> hello
[PC ~/hello]$ mb-flthdr hello ------------------------------------------------------        ②
hello
    Magic:          bFLT
    Rev:            4
    Entry:          0x50
    Data Start:     0x3e60
    Data End:       0x4bf0
    BSS End:        0x6bf0
    Stack Size:     0x1000
    Reloc Start:    0x4bf0
    Reloc Count:    0x53
    Flags:          0x5 ( Load-to-Ram Gzip-Compressed ) ----------------------------        ③
[PC ~/hello]$
```

① Pass a compression option “-z” to “mb-flthdr”.
② Display the header of the executable file generated by the “mb-flthdr” command.
③ The Gzip-Compressed flag can be seen.

11.2.2. Compression when Compiling

To compress when compiling, the FLTFLAGS environment variable is used. The following is an example for the Hello World program.

Example 11-3 Specifying Compression Mode by FLTFLAGS

```
[PC ~/hello]$ make FLTFLAGS=-z   -------------------------------------------------        ①
[PC ~/hello]$ mb-flthdr hello   --------------------------------------------------       ②
hello
      Magic:          bFLT
      Rev:            4
      Entry:          0x50
      Data Start:     0x3e60
      Data End:       0x4bf0
      BSS End:        0x6bf0
      Stack Size:     0x1000
      Reloc Start:    0x4bf0
      Reloc Count:    0x53
      Flags:          0x5 ( Load-to-Ram Gzip-Compressed ) ----------------------------   ③
[PC ~/hello]$
```

① Set the FLTFLAGS environment variable to "-z" and execute "make".
② Display the header of the executable file generated by the "mb-flthdr" command.
③ The Gzip-Compressed flag can be viewed.

11.3. Specifying Stack Size

The following subsections provide information on how to specify stack size. Here, two different methods are shown.

11.3.1. Changing the Stack Size of a Compiled Binary File

Specify stack size with "mb-flthdr –s". While dependant on architecture, the stack size of "4096" is usually the default value.

Example 11-4 Changing Stack Size

```
[PC ~/hello]$ mb-flthdr -s 8192 ---------------------------------------------------     ①
[PC ~/hello]$ mb-flthdr hello ----------------------------------------------------       ②
hello
      Magic:          bFLT
      Rev:            4
      Entry:          0x50
      Data Start:     0x3e60
      Data End:       0x4bf0
      BSS End:        0x6bf0
      Stack Size:     0x2000 -------------------------------------------------------     ③
      Reloc Start:    0x4bf0
      Reloc Count:    0x53
      Flags:          0x1 ( Load-to-Ram )
[PC ~/hello]$
```

① Pass stack size change option "-s" and stack size to "mb-flthdr".
② Display the header of the executable file generated by "mb-flthdr" command.
③ The change to 8192byte can be viewed.

11.3.2. Specifying Stack Size when Compiling

To specify stack size when compiling, the environment variable "FLTFLAGS" is used. An example using the Hello World program is shown below.

Example 11-5 Specifying Stack Size by FLTFLAGS

```
[PC ~/hello]$ make FLTFLAGS='-s 8192' ---------------------------------------------      ①
[PC ~/hello]$ mb-flthdr hello ----------------------------------------------------      ②
hello
      Magic:            bFLT
      Rev:              4
      Entry:            0x50
      Data Start:       0x3e60
      Data End:         0x4bf0
      BSS End:          0x6bf0
      Stack Size:       0x2000 ---------------------------------------------------      ③
      Reloc Start:      0x4bf0
      Reloc Count:      0x53
      Flags:            0x1 ( Load-to-Ram )
[PC ~/hello]$
```

① Set the FLTFLAGS environment variable to “-s 8192” and execute “make”.
② Display the header of the executable file generated using the “mb-flthdr” command.
③ The change to a “8192byte” stack size can be seen.

12. Overwriting Flash Memory

There are three methods available on the SUZAKU to overwrite the Flash memory. This chapter describes these three methods. NetFlash can be used when uClinux is running. Hermit can be used when the SUZAKU is activated in boot loader mode.

12.1. Overwriting Flash Memory Using NetFlash

Overwriting the Flash memory while uClinux is running.

The NetFlash command is used to overwrite the Flash memory. NetFlash downloads an image file via the network and then writes it to the Flash memory. NetFlash supports http, ftp and tftp protocols. To use NetFlash, you will need a server on which an application supporting any one of these protocols is running. If no such a server is available in your environment, please ask your network manager.
As shown in Example 12-1 Activating netflash, the command is issued on the SUZAKU.

Example 12-1 Activating netflash

```
[SUZAKU ~]# netflash http://local.server.name/suzaku_images/image.bin
```

You may change the “local.server.name/suzaku_images/” path to suit your environment.

Once the writing to the Flash memory successfully completes, the system will reboot.

NetFlash provides a wide variety of options. For your reference, the following shows the NetFlash help information.

Example 12-2 netflash Help

```
[SUZAKU ~]# netflash -?
usage: netflash [-bCfFhijklntuv?] [-c <console-device>] [-d <delay>] [-o <offset>]
[-r <flash-device>] [<net-server>] <file-name>

        -b      don't reboot hardware when done
        -C      check that image was written correctly
        -f      use FTP as load protocol
        -F      force overwrite (do not preserve special regions)
        -h      print help
        -i      ignore any version information
        -H      ignore hardware type information
        -j      image is a JFFS2 filesystem
        -k      don't kill other processes (or delays kill until
                after downloading when root filesystem is inside flash)
        -K      only kill unnecessary processes (or delays kill until
                after downloading when root filesystem is inside flash)
        -l      lock flash segments when done
        -n      file with no checksum at end (implies no version information)
        -p      preserve portions of flash segments not actually written.
        -s      stop erasing/programming at end of input data
        -t      check the image and then throw it away
        -u      unlock flash segments before programming
        -v      display version number
[SUZAKU ~]#
```

12.2. Overwriting Flash Memory Using Hermit

This section provides information on how to overwrite the Flash memory using Hermit. Since data is transferred from a PC to the target equipment (SUZAKU in this manual), the operation of overwriting the Flash memory is also referred to as a “download”. The objects which can be overwritten using Hermit are Hermit itself and uClinux. Hermit writes an image file to the specified region or address of the Flash memory via the serial port. For information on the Flash memory map and region, refer to Section 5 Flash Memory Map.

Caution

Make sure that the Hermit is installed. If not, refer to 7.5. Installing Hermit.

12.2.1. Activating in Boot Loader Mode

First, activate the SUZAKU in boot loader mode. Shorting the boot selection jumper and hitting the Enter key on the boot mode selection screen takes the SUZAKU to the boot loader mode. For more information about the boot mode, refer to 6.2. Boot Mode.

12.2.2. Downloading

Enter the following commands from the PC to initiate downloading.

Caution

While the serial port is assigned to Minicom, the downloading always fails since Hermit cannot use the port. Be sure to terminate Mincom before activating Hermit.

To overwrite a uClinux image file (image.bin), use a command similar to Example 12-3 Overwriting image.bin.

Example 12-3 Overwriting image.bin

```
[PC ~]$ hermit download -i ~/uCliunux-dist/images/image.bin -r image
```

To overwrite a Hermit image file (/usr/lib/hermit/loader-suzaku.bin), use a command similar to Example 12-4 Hermit Command.

Example 12-4 Hermit Command

```
[PC ~]$ hermit download -i /usr/lib/hermit/loader-suzaku.bin -r bootloader --force-locked
```

The following is a description of each parameter.

hermit
: Command name of Hermit

download
: Download command

-i file name
: Specify the file to be transferred. “loader-suzaku.bin” is installed together with Hermit.

-r region
  Specify the region where the file is to be transferred to. For information about region, refer to the Flash Memory Map.

--force-locked
  Forcefully write to the locked region.

For your reference, the following shows the Hermit Help information. For more detailed information, refer to Hermit's man page.

Example 12-5 Hermit Help

```
[PC ~]$ hermit
Usage: hermit [options] command [command options]
Available commands: download, help, go, map, terminal, upload
Armadillo-J command: firmupdate
Multiple commands may be given.
General options (defaults) [environment]:
        -e, --ethernet
        -i, --input-file <path>
        --netif <ifname> (eth0) [HERMIT_NETIF]
        --memory-map <path>
        --port <dev> (/dev/ttyS0) [HERMIT_PORT]
        -o, --output-file <path>
        --remote-mac <MAC address>
        -v, --verbose
        -V, --version
Download options:
        -a, --address <addr>
        --force-locked
        -r, --region <region name>
Memory map options:
        --anonymous-regions
[PC ~]$
```

12.3. Overwriting Flash Memory in Motorola S-Format

Downloading in the Motorola S-Format is the final means to update the SUZAKU Flash memory by software. It can be used even when both uClinux and Hermit will not boot. In this manual we provide information on how to download the Motorola S-Format Hermit to the SUZAKU using Minicom.

Motorola S-Format Hermit for both 4MB and 8MB Flash memory is contained in the tools/s-record/ directory of the supplied CDROM under the names of hermit.srec and hermit-8MB.srec. It is assumed that this file is copied under the home directory, and that the 4MB Flash memory is being used is the following examples.

Note

The address to where Hermit will be written to depends on the flash memory size. As the Motorola S-Format files include target address, two separate files must be created as above. Both files are created from the same binary file.

First, connect the SUZAKU to the PC using a serial cross cable and then activate Minicom on the PC. Then short the boot mode jumper on the SUZAKU and turn it on. The boot mode selection screen should then come up on the Minicom screen.

After entering “S” at the boot mode selection screen the following message should be displayed.

Example 12-6 Motorola S-Format Download Initiation Screen

```
Erasing Flash...Done
Start sending S-Record!!
```

After this message is displayed, the SUZAKU waits for the Motorola S-Format file to be transferred.

Select “Send files” from the Minicom menu. Then, select “ASCII” from the Upload submenu and go to the file selection screen. Then, select ”hermit.srec” from the file selection screen to initiate the file transfer. During the file transfer, a small screen appears in Minicom. After the file transfer is complete, the following message is displayed:

```
READY: press any key to continue...
```

Then reboot the SUZAKU. “Hermit” will be activated after the boot loader mode is selected on the rebooted SUZAKU.

13. Troubleshooting

13.1. Hermit

Q. The hermit command cannot be found.

A. Please install Hermit. ⇒ 7.5 Installing Hermit

Q. Hermit produces the following error: “hermit: panic: can't connect to target”.

A. Is the power supply of the SUZAKU on?
Is a serial cross cable properly connected? ⇒ 4.1 Connecting Cables
Is the SUZAKU set to run in download mode? ⇒ 6.2.2 Boot Loader Mode

13.2. Minicom

Q. When I attempt to activate minicom, the following error appears: “command not found”.

A. Install Minicom ⇒ 7.4 Installing Minicom

Q. Even when minicom is activated and the SUZAKU is turned on, nothing appears on the screen.

A. Is the port which Minicom is using the same as the serial port to which the SUZAKU is connected?
Is the serial port to which Minicom is connected configured properly? ⇒
Serial Communication Software

13.3. Compiling uClinux-dist

Q. mb-gcc is not found and results in an error.

A. Install the Tool Chain. ⇒ 7.1 Installing the Tool Chain

Q. Even after the Tool Chain is installed, it still results in a not found error.

A. Is the environment variable “PATH” configured properly? ⇒ 7.2 Setting Environment Variable

Revision History

Rev.	Revision Date	Description of Change
1.0	2004/04/29	First edition of Release
1.0.1	2004/06/04	- Change of Minicom installation from apt to dpkg - Correction to the tool chain name - Generalization of OTC Makefile
1.0.2	2004/12/17	Company address updated
1.1.0	2005/1/31	Added description of a 8MB Flash Memory

SUZAKU Software Manual January 31, 2005 version 1.1.0

Atmark Techno, Inc.
AFT Building 6F, North 5 East 2, Chuo-ku, Sapporo, Hokkaido 060-0035
TEL: 011-207-6550 FAX: 011-207-6570