UM1535 User manual

Getting started with SPEAr® Linux support package (LSP 3.2.5)

Introduction

This manual provides application developers with a first introduction to the Linux-based reference software installed in the Flash memory of the SPEAr evaluation boards. It is not intended to be a tutorial on Linux operating system or embedded software design. It only covers topics that are specific to the implementation on SPEAr embedded MPUs and boards.

The Linux Support Package (LSP) is a set of software provided free-of-charge by STMicroelectronics for the SPEAr family of embedded microprocessor units (eMPUs).

The LSP has the following objectives:

- Demonstrating capabilities of SPEAr devices through a widespread high-level operating system (Linux). The LSP is only targeting STMicro evaluation boards.
- Providing a starting point for customers willing to accelerate a Linux porting to their proprietary SPEAr-based hardware platforms and products.

Contents

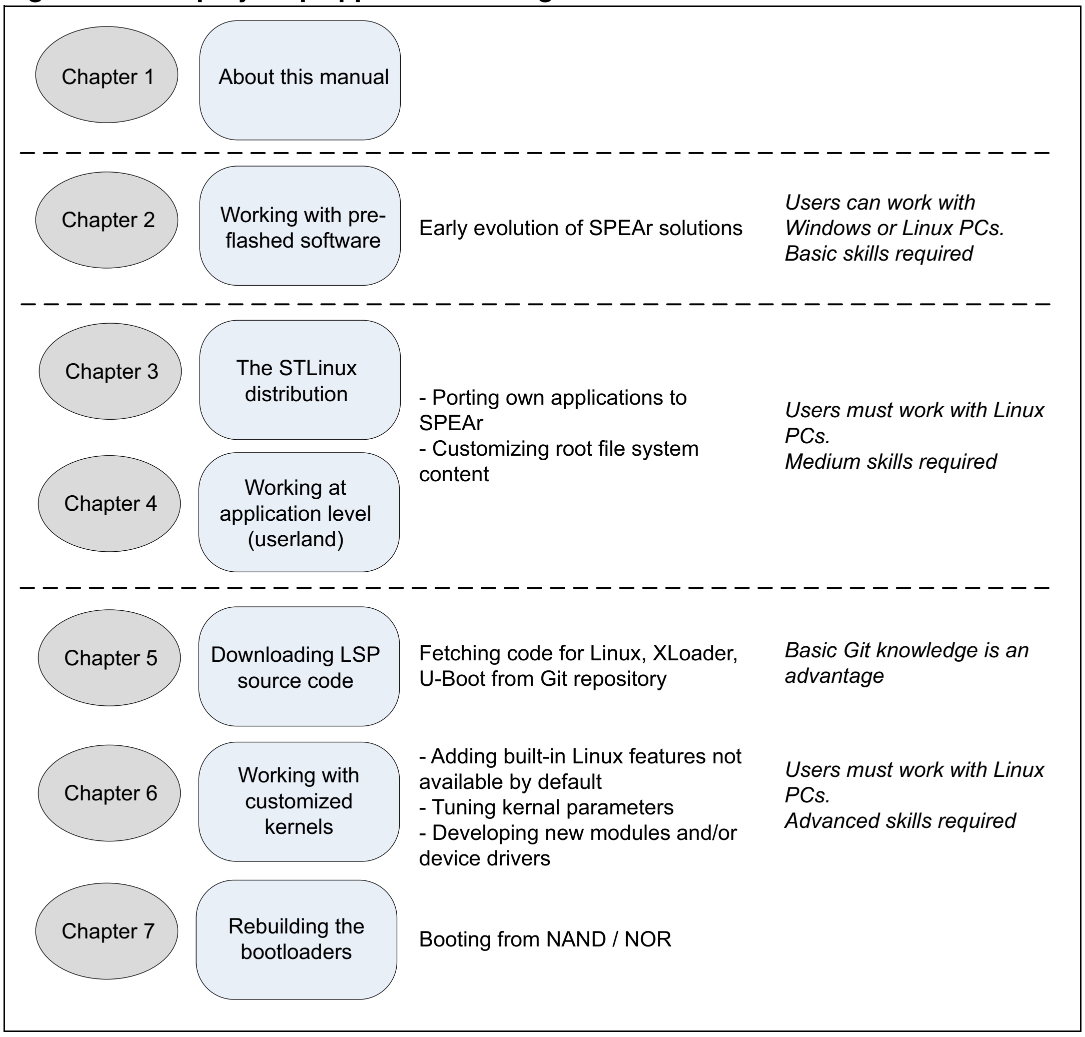
List of tables

List of figures

1 About this manual

This manual focuses on software usage on SPEAr development boards and is organized as a sequence of chapters going from a description of the first step and then covering more complex subjects.

Figure 1. Step-by-step approach to using the manual


Chapter 1
About this manual
Chapter 2
Working with pre-flashed software
Early evolution of SPEAr solutions
Users can work with Windows or Linux PCs. Basic skills required
Chapter 3
The STLinux distribution
Chapter 4
Working at application level (userland)
- Porting own applications to SPEAr
- Customizing root file system content
Users must work with Linux PCs. Medium skills required
Chapter 5
Downloading LSP source code
Fetching code for Linux, XLoader, U-Boot from Git repository
Basic Git knowledge is an advantage
Chapter 6
Working with customized kernels
- Adding built-in Linux features not available by default
- Tuning kernal parameters
- Developing new modules and/or device drivers
Users must work with Linux PCs. Advanced skills required
Chapter 7
Rebuilding the bootloaders
Booting from NAND / NOR

This guide applies to all currently available SPEAr evaluation boards.

However, each different evaluation board is based on a specific member of SPEAr embedded MPU family and provides, in general, a different selection and combination of hardware devices on the board (and companion boards, wherever applicable). For a detailed description of hardware features for each evaluation board, please refer to the corresponding evaluation board user manual.

2 Working with pre-flashed software

SPEAr evaluation boards come with default embedded Linux software already stored in (serial NOR) Flash memory, according to a pre-defined generic configuration. Using a SPEAr board with pre-flashed software is initially useful to get familiar with the target hardware platform and the embedded Linux environment.

This activity does not strictly require the installation of the SPEAr Linux support package (software development environment). This can be performed later as, described in *Chapter 3* of this document.

Before powering-on target hardware, hence booting the pre-flashed software, please carefully check the specific hardware configuration (for example, DIP switches) according to what is described in the relevant hardware manuals.

2.1 Host PC requirements

2.1.1 Windows PC

In order to control the target hardware, use a PC with a Microsoft Windows operating system (XP, Vista, Windows 7).

The first step is to set up a serial port for interacting with the embedded consoles (Linux shell or U-Boot boot loader). If a RS232 serial port is not available on the PC, you can use a USB/RS232 adapter (not provided in the kit).

The second step is to obtain a terminal emulation program. Windows comes with the built-in HyperTerminal, but any equivalent tool can be used as an alternative. For instance, Tera Term is an open source free application with more features and higher flexibility, especially its scripting capability.

You can download and find more technical information about Tera Term on:

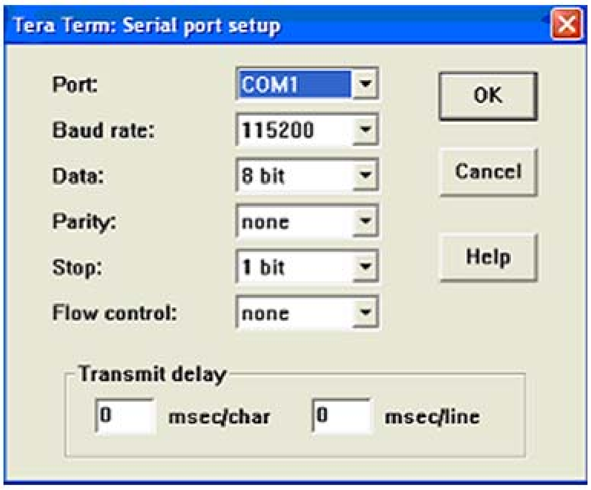
http://en.wikipedia.org/wiki/Tera_Term

In order to configure the serial port with TeraTerm;

1. Launch the tool
2. Click on "Setup > serial port"

   The configuration must reflect that shown in *Figure 2*.
3. To save the proper setting, click on "Setup > save setup".

Figure 2. Tera Term configuration for serial port (Windows PC)


Tera Term: Serial port setup

Port: COM1
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Using HyperTerminal is very similar. To configure the serial port with HyperTerminal:

1. Enter the "File > Properties" menu
2. Select the COM port (for example COM1) in the "Connect Using" dialog box
3. Press the "Configure" button
4. Enter the 'Port Setting' fields accordingly
5. To save the current configurations select the "File >Save As" menu item.

2.1.2 Linux PC

As an alternative to Windows, you can use a PC with Linux OS.

In the examples below, a '$' symbol represents a normal user prompt while a '#' symbol means a root level prompt. Please note that you need read/write access to the PC serial port. If necessary, check your distribution documentation to enable it (for example, on Fedora Linux systems you have usually to add your user to the "dialout" system group).

Minicom is one of the most commonly used terminal emulators for Linux. Assuming a Fedora distribution for the host PC, to check the availability of Minicom, execute the following command:

```
$ rpm -q minicom
```

To install minicom, if not found, execute this command from a root shell:

```
# yum install minicom
```

To start minicom, type the command:

```
$ minicom
```

To enter the configuration menu for the first time, press the key combination "Ctrl-A" and then "Z" (in sequence).

Note: *If there is no global configuration file, minicom will not start. You first need to create one by running the following command from a root shell:*

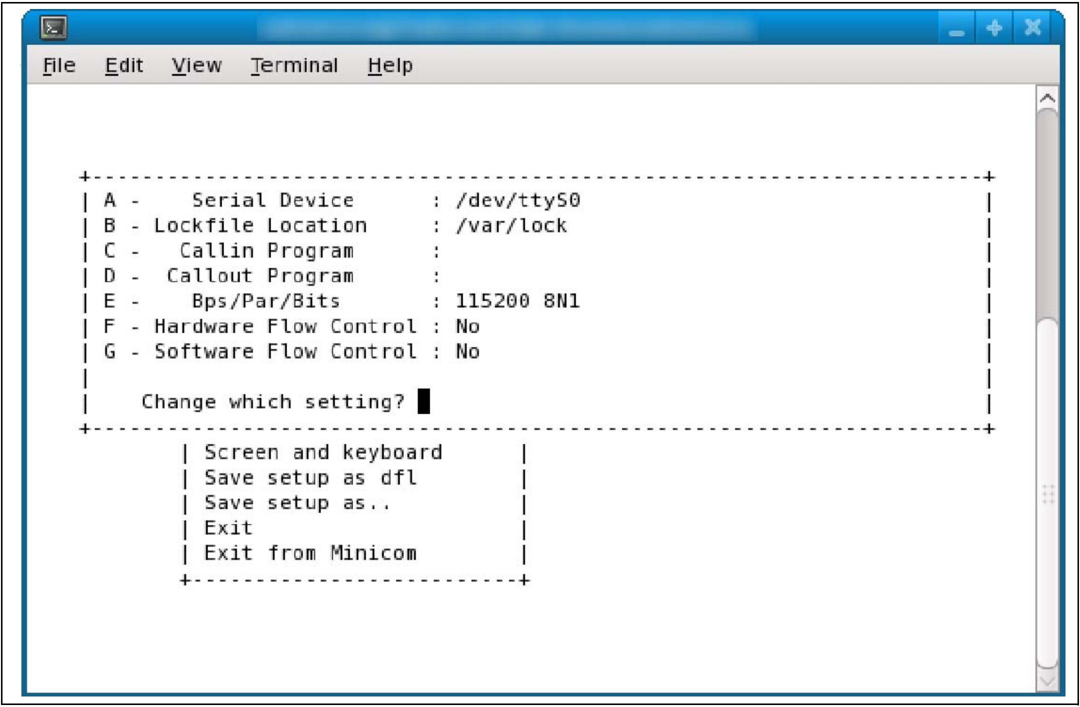
```
# minicom -s
```

and then follow the normal configuration procedure.

The serial connection information can be configured in the 'configure minicom' submenu and then 'Serial port setup'. After that, the configuration must be saved using the 'Save setup as' option. The serial device name to be entered must match the one used for the link to the SPEAr evaluation board. For example, the first serial port on Linux PC is named /dev/ttyS0.

Select the serial speed as 115200 bps with 8 bit, no parity and 1 stop bit (115200 8N1) and disable both hardware and software flow control.

Figure 3. Minicom serial port configuration (Linux PC)



```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyS0                                |
| B - Lockfile Location     : /var/lock                                 |
| 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                     |
        | Exit from Minicom        |
        +--------------------------+
```

To save a new default configuration which is automatically used by minicom, select "Save setup as dfl".

Alternatively, to create a new configuration file select "Save setup as..". In order to use it, you have to specify the configuration file name in the command line when invoking minicom.

Please note that by default minicom tries to initialize a "modem" connected to the specific interface you have chosen. To skip this step you can invoke minicom with the -o option as follows:

```
$ minicom -o
```

2.2 Overview of Flash contents and structure

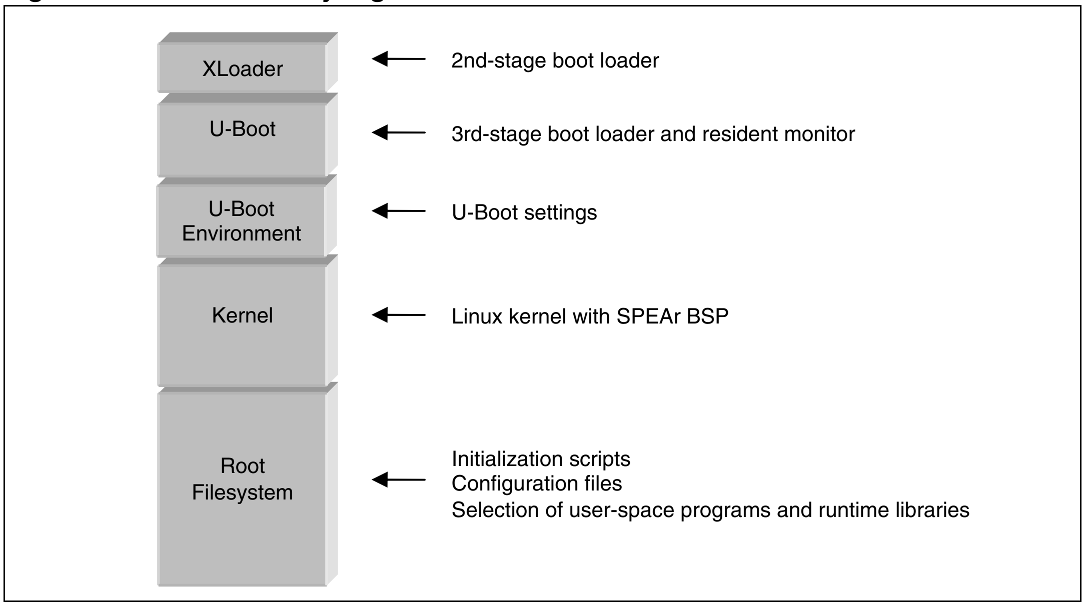
On all evaluation boards, the default software is only pre-stored into serial NOR Flash. The use of other memory types available on some evaluation boards (NAND, parallel NOR) requires the installation and usage of the STLinux distribution, as described in *Chapter 3: The STLinux distribution*.

This subsection only provides an overview of the Flash contents and structuring. Details on this subject (like offsets and sizes for each partition, as well as differences between serial

NOR, parallel NOR and NAND) is provided in the corresponding SPEAr device datasheet and user manual.

Whatever the Flash memory type, software installed in the SPEAr evaluation boards is logically structured into 5 partitions, as depicted in *Figure 4*.

Figure 4. Flash memory organization



To understand the rationale behind this Flash memory organization, it is necessary to know how the overall booting process works.

After power-on, a SPEAr embedded MPU starts to execute an on-chip firmware known as BootROM (kept in internal SPEAr ROM area, not in Flash memory). This is the first stage. When a board is configured in Flash booting mode, BootROM terminates by loading a 2nd stage component (XLoader) from Flash memory to on-chip SRAM and then gives control to it.

XLoader is a small ST-specific firmware (also available in source code) that executes in internal SRAM and configures the PLLs and the specific DDR memory available on each board. Once performed its task, XLoader loads a third stage boot loader (U-Boot) from Flash and jumps to its entry point.

The widely used open source U-Boot program is used as a third-stage boot loader in the SPEAr evaluation boards. The U-Boot version provided is extended with support for the required SPEAr-specific hardware. In addition to its role in the overall booting process, U-Boot also operates as a resident monitor. To start the monitor function after power-on, stop U-Boot execution before Linux is started. U-Boot is mainly in charge of loading and executing the Linux kernel. This stage is configured in a special Flash partition known as U-Boot Environment, basically a collection of parameter/value pairs.

When not interrupted after board power-on, U-Boot loads the Linux kernel to DDR memory and then starts its execution. The kernel, after further preliminary initialization, mounts the so-called root file system. This is the binary image containing all software and data that comprise the operating system complementing the kernel. The root file system is the hierarchical file structure that end users see from the familiar Linux shell prompt. For pre-flashed software, it contains a generic subset of initialization scripts, system commands and runtime libraries.

There is a wide choice of file system standards for Flash memories in the Linux world (for example, CRAMFS, JFFS2, YAFFS2, LogFS, UBIFS).

The pre-flashed software is based on the most commonly used standard, namely JFFS2. The main advantage of JFFS2 is the built-in support for compression. It is also the most consolidated file system, while generally not providing the best performance in terms of booting time.

2.3 Booting up to the Linux prompt

To boot Linux just power up the board. You then see the bootloader messages and prompt. If you press space key in this phase, you stop the bootloader execution and display the U-Boot command prompt (see next section). Instead, wait for a few seconds while the Linux kernel is loaded and launched. Progress is displayed by a sequence of kernel boot messages. The root filesystem is mounted automatically, the system shell is run and a prompt appears when it is ready to accept commands.

The pre-flashed filesystem provides a default set of system commands and runtime libraries.

System commands are the familiar basic utilities mainly provided to support the development and debugging stages. They are all stored under standard Linux paths. They may not be strictly needed in a final product, however their small size allows them to be kept in production devices without a significant penalty in terms of memory footprint.

As is typical for embedded Linux environments, STLinux uses BusyBox, an open source program combining tiny versions of many common user-space Linux utilities into a single small executable, an important factor when minimized Flash footprint is required.

BusyBox has been developed with size-optimization and limited resources in mind, for this reason the available commands typically have fewer options than their full-featured GNU counterparts. However, the most important options are still available with all the functions needed for developing and testing embedded products.

As well as the features of BusyBox, a number of additional executable programs are also included.

In addition to playing with commands, it can be also helpful to explore the Linux standard /proc pseudo-filesystem. This subtree contains user-accessible entries that pertain to the runtime state of the kernel and, by extension, the executing processes that run on top of it. The "pseudo" term is used because the proc filesystem exists only as a reflection of the in-memory kernel data structures it displays. This is why most files and directories within /proc are zero bytes in size.

In practice, the proc file system is intended to be populated at runtime with system information and statistics. Proc files may be either read-only or read-write. Each numerically named directory within /proc corresponds to the process ID (PID) of a process currently executing on the system. This part of the proc file system totally depends on the runtime state of the target. Each numeric entry contains subfiles that provide process-specific information. The other (non-numeric) entries describe some aspects of kernel operation.

Some of the common user commands performed on the proc file system are:

```
# cat /proc/version             Displays full Linux kernel version information

# cat /proc/sys/kernel/osreleaseDisplays Linux kernel release

# cat /proc/sys/kernel/version  Displays Linux kernel build date and time
# cat /proc/cpuinfo          Displays information about the SPEAr CPU
# cat /proc/meminfo          Displays information about memory usage
# cat /proc/modules          Displays information about kernel extension modules
# cat /proc/mtd              Displays information about Flash partitions
# cat /proc/partitions       Displays other information about Flash partitions
# cat /proc/stat             Displays OS status information
# cat /proc/bus/usb/devices  Displays information about USB Host ports
# cat /proc/net/dev          Reports Ethernet information.
# cat /proc/net/tcp          Reports TCP sockets information.
# cat /proc/net/udp          Reports UDP sockets information.
# cat /proc/net/arp          Reports ARP table.
# cat /proc/net/route        Reports IP routing table.
# cat /proc/kallsyms         Lists all kernel symbols
```

The standard /proc/bus/usb subtree is also made available. This is used to access USB Host controllers and plugged devices from user-space applications.

For more details about the functionality provided by pseudo file systems, please refer to standard Linux documentation.

2.4 Using a USB pen drive

USB pen drives can be accessed in a standard Linux way, connecting them to a USB host port and mounting their file system under root file system. An example of operational sequence for a standard pen drive connected as "sda" device (only one pen drive present) is the following:

1. Plug the pen drive in a USB port. and wait for the Linux kernel to autodetect it (you can see active kernel messages on the terminal)
2. Mount the pen drive file system:
   ```
   # mount /dev/sda1 /mnt
   ```
   Now the pen drive file system can be accessed under /mnt directory.
3. Transfer files as usual (for example, cp command).
4. When finished, unmount the pen drive:
   ```
   # umount /mnt
   ```

Now you can physically unplug the pen drive.

2.5 Using a MMC/SD card

Some evaluation boards (currently EVALSP1340CPU / EVALSP320SCPU / EVALSPEAR300 / EVALSPEAR600) provide a MMC/SD card reader slot.

An example of operation sequence for a MMC/SD card is the following:

1. Plug the card into the board slot and wait for the Linux kernel to autodetect it (you can see active kernel messages on the terminal)
2. Mount the card's file system:
   ```
   # mount /dev/mmcblk0p1 /mnt
   ```
   Now the card's file system can be accessed under the /mnt directory.
3. Transfer files as usual (for example, cp command)
4. When finished, unmount the card:
   ```
   # umount /mnt
   ```
5. Now you can physically remove the card.

2.6 Using a SATA hard disk

Some evaluation boards (currently EVALSP1340CPU) support SATA.

An example of the operation sequence for a SATA hard disk is the following:

1. Mount the SATA file system:
   ```
   # mount /dev/sda1 /mnt
   ```
2. Now the SATA file system can be accessed under /mnt directory.
3. Transfer files as usual (for example, cp command).
4. When finished, unmount the SATA:
   ```
   # umount /mnt
   ```
5. Now you can physically unplug the SATA.

2.7 Entering the U-Boot resident monitor

The U-Boot resident monitor offers an interactive command-line interface that can be used through the serial console. U-Boot is executed before starting the Linux OS. In order to interrupt the normal boot process and enter U-Boot operation mode, press a key from the virtual terminal on the Windows or Linux PC during the initial period (after hardware reset) when the following message is displayed:

Hit SPACE to stop autoboot: _

To obtain a full list of U-Boot commands available on SPEAr board, enter the "help" command or simply type the "?" character. When "help" is followed by a command name, a description of that specific command is displayed.

The following subsections describe basic usage scenarios.

Please note that command results are only shown as examples and can look slightly different depending on the evaluation board and software version you use.

Detailed documentation about the described commands, as well as additional ones, may be found on the main U-Boot Web site: http://www.denx.de/wiki/U-Boot.

Note: *Not all the commands mentioned in denx are implemented in U-Boot.*

2.7.1 Information commands

To report the U-Boot version currently available on the evaluation board Flash memory, execute the "version" command:

```
> version u-boot
> version U-Boot 2010.03-lsp-3.2.5 (Feb 22 2012 - 11:31:43)-SPEAr
```

To list all Flash memory partitions, use the "imls" command:

```
u-boot> imls

Legacy Image at E6000000:

   Image Name:   Xloader 2010.06-lsp-3.2.5 for sp

   Image Type:   ARM Linux Firmware (uncompressed)

   Data Size:    27384 Bytes = 26.7 KiB

   Load Address: b3801504

   Entry Point:  b3801504

   Verifying Checksum ... OK

Legacy Image at E6010000:

   Image Name:   U-Boot 2010.03-lsp-3.2.5 for spe

   Image Type:   ARM Linux Firmware (uncompressed)

   Data Size:    196264 Bytes = 191.7 KiB

   Load Address: 00700000

   Entry Point:  00000000

   Verifying Checksum ... OK

Legacy Image at E6050000:

   Image Name:   Linux-2.6.37-lsp-3.2.5-spear13xx

   Image Type:   ARM Linux Kernel Image (uncompressed)

   Data Size:    2579968 Bytes = 2.5 MiB

   Load Address: 00008000

   Entry Point:  00008000

   Verifying Checksum ... OK
```

2.7.2 Environment variables

A specific subgroup of U-Boot commands very often invoked by end users is related to the management of environment variables. Such variables are string-type fields that may be read and written, as well as stored on Flash to guarantee their persistence across system reboots. Some of these variables have a predefined purpose, but users may also add their own custom variables. It is important to know that a current environment maintained in DDR RAM memory can be sometimes different from the persistent environment stored on Flash.

To create a new variable or change the value of an existing one, run the setenv command:

```
> setenv MYVAR=MYVALUE
```

To edit existing variables, use the edit command:

```
> edit MYVAR
> edit: MYVALUE
```

Change the value of the variable MYVAR to a new value, MYNEWVAL and press ENTER.

To make the current values of all variables persistent in Flash memory, run the "saveenv" command:

```
> saveenv
```

Finally, to report all current environment settings, use the "printenv" command:

```
u-boot> print
bootargs=console=ttyAMA0,115200 root=/dev/mtdblock4
rootfstype=jffs2
bootcmd=bootm 0xe6050000
ramboot=setenv bootargs root=/dev/ram rw console=ttyAMA0,115200
$(othbootargs);bootm 0xe6050000
nfsboot=bootp; setenv bootargs root=/dev/nfs rw
nfsroot=$(serverip):$(rootpath)
ip=$(ipaddr):$(serverip):$(gatewayip):$(netmask):$(hostname):$(netd
ev):off
bootdelay=1
baudrate=115200
usbtty=cdc_acm
ethact=mii0
stdin=serial
stdout=serial
stderr=serial
verify=n

Environment size: 507/8188 bytes
u-boot>
```

2.7.3 Advanced commands

More advanced usage of U-Boot includes commands for:

- Reading and writing embedded MPU registers and memory areas
- Executing the contents of an environment variable, handling it as a script (sequence of U-Boot commands)
- Checking the ethernet link between the target board and the host PC (ping)
- Booting the Linux kernel from the ethernet network (by TFTP) instead of using a kernel on Flash memory (bootm, tftp, tftpboot).

Note: Each command has a short name. For example you can type "print" instead of "printenv", as in the previous example, or "savee" instead of "saveenv".

2.8 Connecting the evaluation board to a LAN

The evaluation board should be connected to a developer's host PC over a private LAN or even a point-to-point link. Commonly used private IP addresses (Class C) are in the range of192.168.0.0 - 192.168.255.255.

Note: Some SPEAr evaluation boards (currently EVALSP1340CPU, EVALSP1310CPU, EVALSP320SCPU and EVALSPEAR310) provide multiple Ethernet ports. Each port requires a different IP address. The procedure described below is applicable to a single port, usually the one used for software development.

As an example, let's assume an IPv4 local area network with the following characteristics:

Network IPs:	192.168.1.X
Host PC IP:	192.168.1.1
Evaluation board IP:	192.168.1.10

On a Linux PC, configure the host address as follows:

```
# ifconfig eth0 192.168.1.1 broadcast 192.168.1.255 netmask
255.255.255.0
```

In this example we are assuming that evaluation board is connected to Ethernet port 0 (eth0) of the host machine.

You can configure the IP on the evaluation board side in a static or dynamic way.

For a static configuration, the procedure is as follows:

1. Reset the board and enter U-Boot mode
2. Use the following commands to configure environment variables:
   ```
   > setenv ipaddr 192.168.1.10
   > setenv serverip 192.168.1.1
   > setenv gatewayip 192.168.1.1
   > setenv netmask 255.255.255.0
   > setenv hostname SPEAR
   ```
3. Set the root path as path of your NFS file system. If you are using STLinux2.4 as NFS file system then your root path will be:
   – For Armv5 SoCs:
   ```
   > setenv rootpath /opt/STM/STLinux-2.4/devkit/armv5/target
   ```
   – For Armv7 SoCs:
   ```
   > setenv rootpath /opt/STM/STLinux-2.4/devkit/armv7/target
   ```
   Default bootargs contain an environment variable named nfsboot. This variable uses the environment variables above. If extra information is required, the environment variable othbootargs can be used.
4. As final step, modify bootargs with nfsboot. Execute the following command for this
   ```
   > run nfsboot
   ```

In this way, network settings work inside U-Boot and are also automatically passed to Linux.

The dynamic configuration requires a DHCP server running on the LAN (for example, on a Linux host PC) so that the evaluation board automatically gets an IP address at each bootstrap.

To check if the DHCP server (daemon) support is available on the PC-side, use the following command:

```
$ rpm -q dhcp
```

If the DHCP package is not available, install it on the PC from your distribution media (for example, Fedora Linux CDROM or website).

The next step is to create or change the DHCP configuration file /etc/dhcp/dhcpd.conf that matches a specific network setup, as shown in the following example:

```
#
```

```
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#   see 'man 5 dhcpd.conf'
#
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
allow bootp; #allows kernel download using tftp
ddns-update-style ad-hoc;
subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.1;
  option subnet-mask 255.255.0.0;
  option domain-name "local.net";
  option domain-name-servers ns.local.net;
  host SPEAr {
    hardware ethernet 00:11:22:33:44:55;
    fixed-address 192.168.1.10;
    option host-name "SPEAR";
    next-server       192.168.1.1;
    filename "uImage.img"; #kernel image filename in tftp path
    option root-path /opt/STM/STLinux-2.4/devkit/armv7/target/
    }
}
```

With this configuration, the DHCP server replies to a request from a SPEAr evaluation board with Ethernet MAC address 00:11:22:33:44:55 (just as in the example) by passing the entire network configuration.

The "allow bootp" and "filename" parameters enable U-Boot to download and boot a kernel image from the host using the TFTP file transfer protocol (you will need a TFTP server on the host, see below).

In order to apply the new configuration settings, restart he DHCP server. To do this, first check the status of the DHCP daemon using the following commands from the Linux PC shell:

```
# /etc/rc.d/init.d/dhcpd status
```

If the DHCP daemon is stopped, start it with the following command:

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

Please be sure to disable your firewall or setup new firewall rules in order to enable DHCP/TFTP traffic.

2.9 Updating the pre-flashed software

This section focuses on updating the contents of serial NOR Flash with default binary images available on ST Web site. It also focuses on updating NAND Flash.

Updating the contents of Flash memory on SPEAr evaluation boards is a recommended step. It is possible to update the pre-flashed software delivered by default with a SPEAr evaluation board without installing the STLinux distribution.

The procedure for the update involves the installation and use of the USB Flasher tool provided by ST.

This tool is a graphical interactive application, which can be downloaded from ST company site, which also provides download for updated binary images (for each evaluation board type) to be used with the USB Flasher. The use of the USB Flasher for other purposes and scenarios is explained in the help pages embedded in the tool itself.

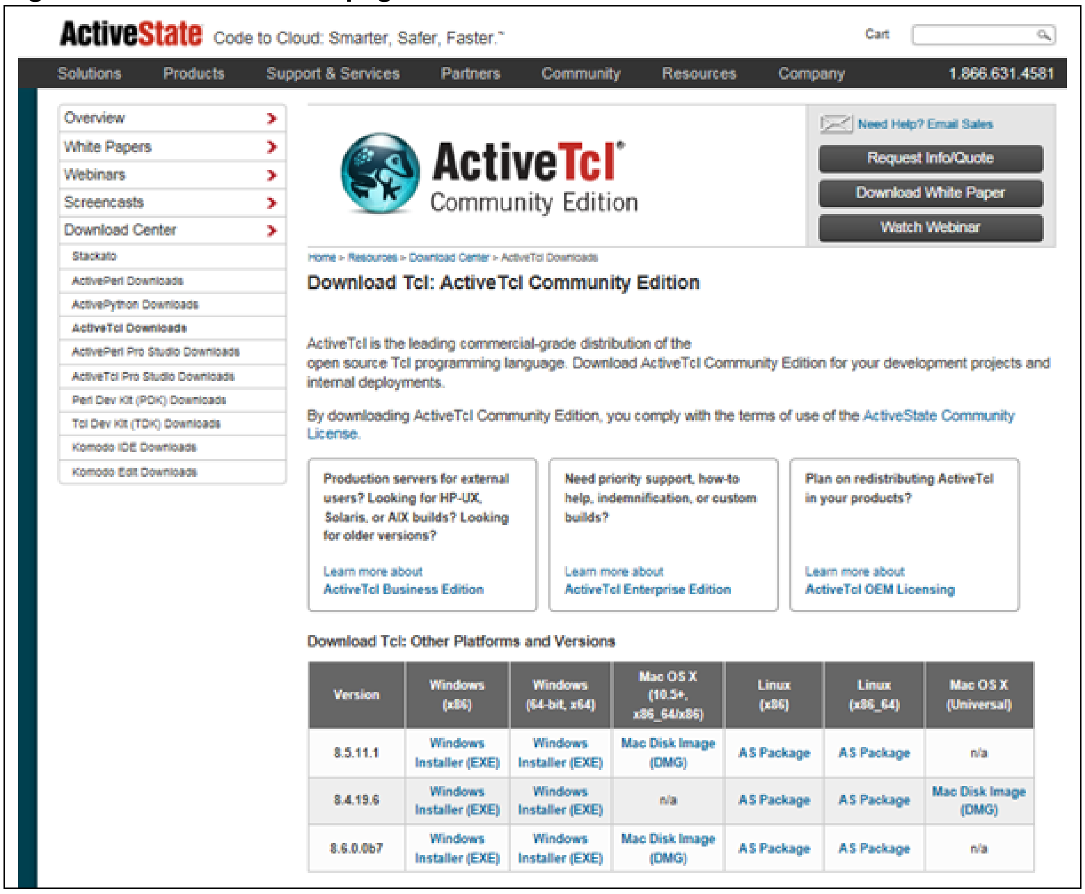
2.9.1 Installing the USB Flasher

The USB flasher tool must be installed on a Windows PC.

Before setting up the flasher tool, download and install an updated TCL package. The recommended TCL software is ActiveTCL, available free of charge. The latest TCL version (8.5) can be downloaded from the following supplier site:

http://www.activestate.com/activetcl/

Figure 5. TCL download page



ActiveState Code to Cloud: Smarter, Safer, Faster.™ Cart

Solutions Products Support & Services Partners Community Resources Company 1.866.631.4581

- Overview
- White Papers
- Webinars
- Screencasts
- Download Center
- Stackato
- ActivePerl Downloads
- ActivePython Downloads
- ActiveTcl Downloads
- ActivePerl Pro Studio Downloads
- ActiveTcl Pro Studio Downloads
- Perl Dev Kit (PDK) Downloads
- Tcl Dev Kit (TDK) Downloads
- Komodo IDE Downloads
- Komodo Edit Downloads

ActiveTcl® Community Edition

Need Help? Email Sales

Request Info/Quote

Download White Paper

Watch Webinar

Home » Resources » Download Center » ActiveTcl Downloads

Download Tcl: ActiveTcl Community Edition

ActiveTcl is the leading commercial-grade distribution of the open source Tcl programming language. Download ActiveTcl Community Edition for your development projects and internal deployments.

By downloading ActiveTcl Community Edition, you comply with the terms of use of the ActiveState Community License.

Production servers for external users? Looking for HP-UX, Solaris, or AIX builds? Looking for older versions?
Learn more about ActiveTcl Business Edition

Need priority support, how-to help, indemnification, or custom builds?
Learn more about ActiveTcl Enterprise Edition

Plan on redistributing ActiveTcl in your products?
Learn more about ActiveTcl OEM Licensing

Download Tcl: Other Platforms and Versions

Version	Windows (x86)	Windows (64-bit, x64)	Mac OS X (10.5+, x86_64/x86)	Linux (x86)	Linux (x86_64)	Mac OS X (Universal)
8.5.11.1	Windows Installer (EXE)	Windows Installer (EXE)	Mac Disk Image (DMG)	AS Package	AS Package	n/a
8.4.19.6	Windows Installer (EXE)	Windows Installer (EXE)	n/a	AS Package	AS Package	Mac Disk Image (DMG)
8.6.0.0b7	Windows Installer (EXE)	Windows Installer (EXE)	Mac Disk Image (DMG)	AS Package	AS Package	n/a

The Flashing utility comes with details on installing and using the Flashing utility. This document can be found at: doc\ html\ flasher.htm.

Follow instructions in flasher.htm to install the Flashing utility.

3 The STLinux distribution

3.1 Overview

The STLinux distribution provides all the host-side (PC) and target-side (evaluation board) software components enabling system designers to develop their own applications for SPEAr-based platforms, as well as customize the various aspects of the embedded software architecture.

The components can be summarized as follows:

- Command-line cross-development toolchain (compiler, linker, building tools, etc.), running on a Linux x86 PC
- Graphical IDE (ST Workbench), running on a Linux x86 PC
- Software for incremental online update of STLinux distribution components
- A set of open source user-space ARM packages (programs and runtime libraries) to be promptly reused in root file systems as support to specific applications. The root file system also provides compiler, linker and so on for compiling applications on target boards.
- Linux kernel (2.6.37 or higher), configurable for the different SPEAr evaluation boards; it includes BSP/device drivers for the SPEAr hardware features
- U-Boot boot loader, with added support for SPEAr evaluation boards
- XLoader firmware, configurable for the different SPEAr evaluation boards

Most distribution components are also available as source code (SRPM files).

Full details about STLinux components are reported in other specific documents.

3.2 Host PC requirements

The STLinux-2.4 distribution is supported on Fedora 6 and later and Redhat Enterprise 4 and later.

STLinux-2.4 is not officially supported on Ubuntu. The guide for installing on an Ubuntu machine is available at following link:

http://www.stlinux.com/install/ubuntu

3.3 The stmyum tool

Yum is an automatic updater and package installer/remover for RPM Linux host systems (like Fedora). It automatically computes dependencies and figures out the procedure for installing packages.

Yum has a number of advantages over using RPM directly:

- automatic updates of installed packages
- automatic checks for newly available and updated packages
- all dependent packages are installed automatically for each package installed
- groups of packages can be installed as a single unit
- increased security as all packages are signed

The STLinux distribution relies on a slightly customized version of Yum, known as stmyum, to ensure that it does not interfere with any native Yum installation. It is automatically installed at /opt/STM/STLinux-X.X/host/bin and the PATH environment variable must include this directory to be able to use it.

3.4 Installing the STLinux distribution

Installing from the from the STLinux.com website

To install the STLinux distribution from the STLinux.com website, perform the following steps:

1. Go to the download section of STLinux
2. Download section points to ftp site of STLinux (ftp://ftp.stlinux.com/pub/stlinux/)
3. Click the 2.4 link on this web site
4. Download install script on your Linux PC
5. Execute the following command to install STLinux-2.4:

   ARMV7
   ```
   ./install all-armv7-glibc-spear
   ```
   ARMV5
   ```
   ./install all-armv5-glibc-spear
   ```
6. The STLinux tool chain installs at the following path: /opt/STM/STLinux2.4/

   If you have STLinux2.3 installed on you machine, they can co-exist.
7. When the installation is completed, configure the Linux shell environment as follows:

   ARMV7:
   ```
   $ export PATH=$PATH:.: /opt/STM/STLinux-2.4/devkit/armv7/bin
   $ export CROSS_COMPILE=armv7-linux-
   ```
   ARMV5:
   ```
   $ export PATH=$PATH:.: /opt/STM/STLinux-2.4/devkit/armv5/bin
   $ export CROSS_COMPILE=armv5-linux-
   ```

As well as these paths, the host binaries path is added with `$ export PATH=$PATH:.: /opt/STM/STLinux-2.4/host/bin`.

Note: In order to preserve this configuration, the same commands should be also be put in a bash shell initialization file.

Installing from FTP (rsync command)

The STLinux-2.4 distribution can also be installed by copying contents from the FTP site and then installing.

Using the rsync command makes copying and synching easy.

The same command can be run periodically to keep the local copy in sync with the updates on the remote server. These local copies can be used to install the STLinux distribution.

ARMV7
```
./install all-armv7-glibc-spear
```
ARMV5
```
./install all-armv5-glibc-spear
```

1. Run following command

   ```
   rsync rsync://ftp.stlinux.com
   ```

   Sub-repositories are listed, the important ones being:

stlinux2.4-armv5	STLinux 2.4 ARM v5 RPMS
stlinux2.4-armv7	STLinux 2.4 ARM v7 RPMS
stlinux2.4-SRPMS	STLinux 2.4 SRPMS

2. Do a local copy on a single sub-repository with:

   ```
   rsync -tav rsync://ftp.stlinux.com/stlinux2.4-armv7/ ./my_copy
   ```

   Make sure that you add "/" at the end of the remote path, but not at the end of the local path.
3. Install the STLinux-2.4 workbench with the following command:

   ```
   ./install stworkbench
   ```

4 Working at application level (userland)

4.1 Workflow models

When working at application level (the so-called "userland") developers are only concerned with programs and libraries stored in a root file system. Bootloaders and the kernel are assumed to be stable and stored in Flash memory.

There are many approaches (workflows) to modify/extend the root file system for specific application scenarios; the main ones are described in the following subsections.

4.1.1 Remote mounting of the root file system (NFS)

If the board can be connected to the development PC through a Ethernet LAN, the most practical solution is to leave the root file system stored on the PC and remotely mount it on the target embedded Linux OS through the NFS protocol.

The advantages of this approach are:

- The root file system has no global size constraints. Developers can keep hundreds or thousands of packages (programs and libraries) in a directory of their PC disk. All file access from the Linux OS running on the board is performed over the network in a transparent way. Files are not copied to Flash memory, but loaded to DDR RAM strictly on demand.
- A program or library can be simply built (compiled and linked) with the output file on the PC disk; the new version is then available for execution on the board without any need for manual transfer or board reboot.

The drawbacks are:

- File access by NFS over LAN can be slower than direct Flash memory access
- There is no early assessment of which files are actually used and of the overall required size for future migration to Flash memory

To remotely mount the root file system, configure and start the NFS server on Linux PC.

Assuming the NFS server functionality is already provided by your host, the only configuration is an entry for your target root directory to your /etc/exports file, for example:

For ARMV7

```
opt/STM/STLinux-2.4/devkit/armv7/target 192.168.0.0/24
(rw,no_root_squash,sync)
```

For ARMV5

```
/opt/STM/STLinux-2.4/devkit/armv5/target 192.168.0.0/24
(rw,no_root_squash,sync)
```

This line exports the /opt/STM/STLinux-2.4/devkit/armvx/target directory with read and write permission to all hosts on the 192.168.0.0 subnet.

To check NFS availability and start the services, use the following commands (from user root account):

```
# rpm -q nfs-utils
```

After modifying the /etc/exports file, make sure the NFS system is notified about the change, for example by running the command:

```
# service portmap start
Starting portmap: [ OK ]
```

followed by the command:

```
# service nfs start
Starting NFS services: [ OK ]
Starting NFS quotas: [ OK ]
Starting NFS daemon: [ OK ]
Starting NFS mountd: [ OK ]
```

Each time you change this file when the NFS service is already started, you need either to restart it or to force the NFS daemon to reload the new configuration:

```
# exportfs -a
```

4.1.2 Incremental changes to Flash file system

If the root file system is stored on Flash memory, it is possible to transfer files from the PC to the target board in the following ways:

- Transfer by USB pen drive
- Transfer by MMC/SD card
- Transfer by LAN (TFTP)

4.1.3 Flash file system full replacement

To replace the entire file system on Flash memory, you can:

- Rewrite the root file system partition with the USB Flasher tool
- Rewrite the root file system partition from U-Boot
- Rewrite the root file system from Linux by overwriting the mtdblock of the file system

4.1.4 Setting environment variables for using NFS boot

The following U-Boot commands set up environment variables for using the NFS file system:

```
u-boot>setenv bootargs root=/dev/nfs rw console=ttyAMA0,115200
nfsroot=192.168.1.1:/opt/STM/STLinux-2.4/devkit/armv7/target,udp
ip=192.168.1.10:192.168.1.1:192.168.1.1:255.255.255.0::eth0
```

These commands exports the /opt/STM/STLinux-2.4/devkit/armv7/target directory from the host machine (IP address 192.168.1.1) to the target board (IP address 192.168.1.10); the gateway address used is 255.255.255.0.

4.2 Command line cross-development

The most important item concerning host packages is the cross-development toolchain, a set of programs running on a host PC, but targeting ARM-specific code output with support for:

- Cross-compilation of source code to generate native object code for the ARM CPU cores integrated into SPEAr embedded MPU family
- Cross-linking of ARM object code to generate executable programs or (dynamically linkable) shared libraries
- Managing object code archives, incremental rebuilding and other auxiliary tasks

The provided toolchain is based on the widely adopted open source GNU toolset. A summary of the main available command-line tools is reported in *Table 1*. For detailed documentation please consult the GNU Web site: http://www.gnu.org/manual/manual.html.

Table 1. Main toolchain commands

Package	Tool	Description
GCC	Gcc	C Cross-compiler for ARM
	Gcov	Code coverage
binutils	Ar	Archiver
	As	Cross-assembler for ARM
	Gprof	Profiling tool
	Ld	Cross-linker for ARM
	Nm	Lists symbols in object files
	Objcopy	Copies a binary file.
	Objdump	Displays information from object files.
	Ranlib	Generates an index to speed access to archives.
	Readelf	Displays the information about the contents of ELF format files
	Rtrip	Removes symbols and sections from files
GNU Make	Make	Incremental build management
GDB	Gdb	Debugger

4.3 Userland packages

The general STLinux distribution comes with a set of user-space packages prebuilt for the ARM target. This set includes programs and libraries for:

- Graphics: for example, X Server, SDL, Cairo, DirectFB, GTK+, Qt Embedded
- Programming language runtimes: for example, micro Perl
- Connectivity: for example, usblib, Bluetooth (Bluez)
- Multimedia frameworks: for example, ALSA, GStreamer
- Database: for example, sqlite
- Benchmarking tools: for example, IOZONE, netperf

Some of these packages are not supported on SPEAr evaluation boards yet, in particular ones requiring specific hardware for audio, video, wireless connectivity.

4.4 Rebuilding the root file system

The default root file system provided as pre-flashed on all SPEAr evaluation boards, as well as binary image on ST Web site for upgrades, is built for serial NOR Flash and formatted according to JFFS2 structure.

In order to rebuild this configuration, the procedure to be run on Linux host PC is as follows:

```
$ mkfs.jffs2 -n -p -l -s 0x200 -e 0x10000 -r $BUSYBOX/_install/ -o
rootfs_nor.img
```

The same procedure and results can be directly used in case of parallel NOR Flash, always assuming JFFS2. Note that there is no dependency on the specific SPEAr embedded MPU or evaluation board.

For evaluation boards provided with NAND Flash, use the JFFS2 file system.

In order to rebuild a JFFS2 root file system for NAND, on the Linux host PC run:

```
$ mkfs.jffs2 -n -p -l -s 0x200 -e 0x4000 -r $BUSYBOX/_install/ -o
rootfs_nand_smallpage.img
$BUSYBOX/_install/
```

5 Downloading LSP source code

There are two options for downloading the LSP source code:

- From the STLinux Web site

  STLinux distribution contains rpm for source code for XLoader, U-Boot and Linux.

  The default installation of STLinux2.4 source code can be retrieved from:

  /opt/STM/STLinux2.4/devkit/sources

- From the GIT repository

  XLoader, U-Boot and Linux kernel development for SPEAr is done on GIT.

  ST GIT repositories are:

  – Linux: http://git.stlinux.com/?p=spear/linux-2.6.git
  – U-Boot: http://git.stlinux.com/?p=spear/u-boot.git;
  – XLoader: http://git.stlinux.com/?p=spear/xloader.git;

  Source code can be viewed on this link using any internet browser.

  GIT source code can obtained using GIT. Following is an example for kernel source. An identical procedure can be used for XLoader and U-Boot.

1. Clone the repository or fetch new code. If you are downloading SPEAr kernel source for the first time, you should clone the repository. This can be done with:

   ```
   git clone git://git.stlinux.com/spear/xloader.git
   ```

Note: If you are running the command inside a company Intranet, you may be required to set the http_proxy (or HTTP_PROXY) variable with the DNS name of the proxy machine:

```
# export htpp_proxy=http://<yourlogin>:<yourpwd>@<yourproxyaddr>:8080
```

   If you have already cloned the repository, you can fetch updates with the following command:

   ```
   git fetch origin
   ```

2. Create a new branch from the release.

   After cloning you can create your branch from the tag of the LSP release.

   For example, to retrieve code for LSP 3.2.5, execute the following command:

   ```
   git checkout -b <your_branch> lsp-3.2.5
   ```

   This creates your branch with objects from LSP 3.2.5

Note: Information on tags regarding all releases are mentioned in corresponding release notes.

6 Working with customized kernels

When working with SPEAr evaluation boards, making modifications to supplied Linux kernel is only needed when:

- Changing kernel configuration, by enabling or disabling some options or features
- Developing new drivers on top of existing ones, in order to interface peripherals that can be added through custom add-on boards connected to SPEAr evaluation boards (where applicable)
- Rewriting partially or totally some existing reference device drivers (for example, for further optimization or special needs)
- Developing custom kernel modules

The use of kernel-level software with hardware platforms other that SPEAr evaluation boards is not discussed in this section.

When working at kernel level, developers are concerned with the kernel source code tree from which a single binary image file must be generated by "rebuilding" the kernel.

The main tasks to be performed are:

- Kernel reconfiguration
- Kernel rebuild
- Kernel loading and execution on a target evaluation board (different possible workflows)

For general information about Linux kernel, please refer to public Linux documentation.

For the path to the LSP source code, please see the release notes.

6.1 Reconfiguring the kernel

The kernel configuration is managed with standard Linux kernel configuration tools, like "make menuconfig".

Each SPEAr evaluation board has a default kernel configuration file in the kernel source tree directory arch/arm/configs, as shown in following table:

Table 2. Kernel configuration files

Evaluation board	Kernel configuration file
EVALSP1340CPU / EVALSP1310CPU	Spear13xx_defconfig
EVALSPEAR300 /EVALSPEAR310 / EVALSP320SCPU	Spear3xx_defconfig
EVALSPEAR600	Spear6xx_defconfig

To configure the kernel using a default configuration (for example, for SPEAr1340), enter the kernel source tree directory and run the commands:

```
$ make distclean
$ make ARCH=arm CROSS_COMPILE=armv7-linux- spear13xx_defconfig
```

To modify the default configuration, from the terminal run the make ARCH=arm CROSS_COMPILE=armv7-linux- menuconfig tool in the kernel source tree root directory. Now you can change all the configuration options.

Use the arrows keys to navigate in the menu structure and change the options. When all changes are done, save the configuration and exit. The new configuration is saved in a hidden ".config" file. If you like, you can choose another name.

6.2 Rebuilding the kernel

After the configuration step is finished, the kernel can be rebuilt with a "make" command:

```
$ make ARCH=arm CROSS_COMPILE=armv7-linux- uImage
```

If you configured some components as modules, you also need a make ARCH=arm CROSS_COMPILE= armv7-linux- modules command afterwards.

The time required for the build process is varies depending on the number of features you choose during the configuration step and your PC speed (usually it takes some minutes).

When the build process is completed, you can find the kernel images in the arch/arm/boot subdirectory of the kernel tree, both as uncompressed (Image) and compressed (zImage, uImage) kernel image.

6.3 Workflow models

The kernel binary image is always generated on the development PC. However, there are many approaches (workflows) to make the kernel operating on the target evaluation board. The main ones are described in the following subsections.

6.3.1 Booting the kernel on-demand

If the board can be connected to the development PC through an Ethernet LAN, one solution is to load the kernel binary image file just after reset by configuring U-Boot for automatic file transfer by TFTP protocol.

This approach is especially useful when frequently modifying the kernel, for instance during adaptations to a device drivers that are statically linked with kernel code.

To enable TFTP support on the development PC, you must make sure that the TFTP daemon program /usr/sbin/in.tftpd is installed. On Fedora Core systems you can verify this by running:

```
$ rpm -q tftp-server
```

If necessary, install the TFTP daemon program from your distribution media.

Most Linux distributions disable the TFTP service by default. To enable it, for example on Fedora systems, edit the file /etc/xinetd.d/tftp and remove the line

```
disable = yes
```

or change it into a comment line by putting a hash character in front of it. An example of TFTP file configuration is shown below:

```
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -c -v -s /tftpboot
        disable                 = no
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

Also, make sure that the /tftpboot directory exists and is word-readable (permissions at least "dr-xr-xr-x"), and remember to disable the firewall if present (or make new rules to allow TFTP traffic).

The firewall can be disabled as in the following example command:

```
# /etc/init.d/iptables stop
```

Please note that server_args keeps the default search path for TFTP daemon. This is useful for specifying file names omitting this path on the TFTP client side (target). After this, restart the xinetd daemon to force it to reload the new configuration. Type the following command from the host PC prompt:

```
# service xinetd restart
Stopping xinetd: [OK]
Starting xinetd: [OK]
```

6.3.2 Updating the kernel on Flash memory by U-Boot

You can use U-Boot and TFTP transfer to write a new kernel image on Flash memory.

For example, use the following procedure to update the kernel on serial NOR Flash for the SPEAR1300EVAL evaluation board:

1. Press the space key to enter U-Boot autoboot
2. Configure network parameters (statically or with DHCP, see relevant sections)
3. Start the TFTP server on the host machine
4. Place the desired image in the host PC TFTP root directory (for example, /tftpboot/uImage_new.img)
5. From the U-Boot console, download the image to RAM (here we use address 0x0) with TFTP client:
   ```
   > tftp 0x0 uImage_new.img
   ```
6. Erase the specified kernel area on serial NOR:
   ```
   > erase 0:5-31
   ```
   Write the image on serial NOR (you need to know the image size). In this example we copied 1663104 bytes (0x196080 in hexadecimal) to the start of the kernel area on serial NOR (0xF8050000 in hexadecimal):
   ```
   > cp.b 0x0 0xF8050000 0x196080
   ```
7. Reboot the evaluation board

7 Rebuilding the bootloaders

7.1 XLoader

The default XLoader provided as pre-flashed on all SPEAr evaluation boards, as well as binary image on ST Web site for upgrades, is built for serial NOR Flash.

If needed, the XLoader firmware image can be rebuilt. Relevant commands to be run on Linux PC are described below.

EVALSPEAR300

To rebuild U-Boot on a Linux host PC, run the following commands:

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear300_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSPEAR310

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear310_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP320SCPU + PLC Expansion board

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear320_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP320SCPU + HMI Expansion board

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear320_hmi_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSPEAR600

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear600_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP1310CPU

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv7-linux- spear1310_reva_config
$ make ARCH=arm CROSS_COMPILE=armv7-linux-
```

EVALSP1340CPU

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv7-linux- spear1340_config
$ make ARCH=arm CROSS_COMPILE=armv7-linux-
```

7.2 U-Boot

The default U-Boot provided as pre-flashed on all SPEAr evaluation boards, as well as binary image on ST Web site for upgrades, is built for serial NOR Flash.

If needed, the U-Boot firmware image has to be rebuilt taking into account each specific embedded MPU and Flash memory type. Therefore, there are many cases and relevant commands as described below.

EVALSPEAR300

To rebuild U-Boot on a Linux host PC, run the following commands:

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear300_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSPEAR310

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear310_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP320SCPU + PLC Expansion Board

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear320_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP320SCPU + HMI Expansion Board

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear320_hmi_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSPEAR600

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv5-linux- spear600_config
$ make ARCH=arm CROSS_COMPILE=armv5-linux-
```

EVALSP1340CPU

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv7-linux- spear1300_config
$ make ARCH=arm CROSS_COMPILE=armv7-linux-
```

EVALSP1310CPU

```
$ make mrproper
$ make ARCH=arm CROSS_COMPILE=armv7-linux- spear1310_config
$ make ARCH=arm CROSS_COMPILE=armv7-linux-
```

8 Glossary

Table 3. List of abbreviations

Term	Definition
API	Application programming interface
ARM	Advanced RISC machine
BSP	Board support package
DDR	Double data rate (RAM)
DHCP	Dynamic host configuration protocol
FAT	File allocation table
FTP	File transfer protocol
GCC	GNU compiler collection
GPL	General public license (GNU)
IDE	Integrated development environment
IP	Internet protocol
LAN	Local area network
LGPL	Lesser GPL
LSP	Linux support package
MAC	Media access control
MTD	Memory technology device
NFS	Network file system
OS	Operating system
RAM	Random access memory
RPM	RPM package manager
RTC	Real time clock
SDK	Software development kit
SRAM	Static RAM
TFTP	Trivial file transfer protocol
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus
ST site	http://www.stlinux.com/drupal/csd

9 Software license agreement

This Software License Agreement ("Agreement") is displayed for you to read prior to downloading and using the Licensed Software. If you choose not to agree with these provisions, do not download or install the enclosed Licensed Software and the related documentation and design tools. By using the Licensed Software, you are agreeing to be bound by the terms and conditions of this Agreement. Do not use the Licensed Software until you have read and agreed to the following terms and conditions. The use of the Licensed Software implies automatically the acceptance of the following terms and conditions.

9.1 Definitions

Licensed Software: means the enclosed demonstration software and all the related documentation and design tools licensed in the form of object and/or source code as the case maybe.

Product: means a product or a system that includes or incorporates solely and exclusively an executable version of the Licensed Software and provided further that such Licensed Software executes solely and exclusively on ST products.

9.2 License

STMicroelectronics ("ST") grants you a non-exclusive, worldwide, non-transferable (whether by assignment, law, sublicense or otherwise), revocable, royalty-free limited license to: (i) make copies, prepare derivatives works, display internally and use internally the source code version of the Licensed Software for the sole and exclusive purpose of developing executable versions of such Licensed Software only for use with the Product; (ii) make copies, prepare derivatives works, display internally and use internally object code versions of the Licensed Software for the sole purpose of designing, developing and manufacturing the Products; (iii) make, use, sell, offer to sell, import or otherwise distribute Products.

9.3 Ownership and copyright

Title to the Licensed Software, related documentation and all copies thereof remain with ST and/or its licensors. You may not remove the copyrights notices from the Licensed Software. You may make one (1) copy of the Licensed Software for back-up or archival purposes provided that You reproduce and apply to such copy any copyright or other proprietary rights notices included on or embedded in the Licensed Software. You agree to prevent any unauthorized copying of the Licensed Software and related documentation.

9.4 Restrictions

Unless otherwise explicitly stated in this Agreement, You may not sell, assign, sublicense, lease, rent or otherwise distribute the Licensed for commercial purposes, in whole or in part purposes (unless you are an authorized ST distributor provided that all the other clauses of this DEMO PRODUCT LICENSE AGREEMENT shall apply entirely).

You acknowledge and agree that any use, adaptation translation or transcription of the Licensed Software or any portion or derivative thereof, for use with processors manufactured by or for an entity other than ST is a material breach of this Agreement and requires a separate license from ST.

No source code and/or object code relating to and/or based upon Licensed Software is to be made available by You to any third party for whatever reason.

You acknowledge and agrees that the protection of the source code of the Licensed Software warrants the imposition of security precautions and You agree to implement reasonable security measures to protect ST's proprietary rights in the source code of the Licensed Software. You shall not under any circumstances copy, duplicate or otherwise reproduce the source code of the Licensed Software in any manner, except as reasonably necessary to exercise Your's rights hereunder and make one back-up copy. You are granted the right to make one archival or backup copy of the source code of the Licensed Software, which copy shall be marked as an archival copy and as the confidential information of ST. Access to the source code of the Licensed Software shall be restricted to only those of Your employees with a need-to-know for the purpose of this Agreement. You will not under any circumstances permit the source code of the Licensed Software in any form or medium (including, but not limited to, hard copy or computer print-out) to be removed from your official premises as you have informed us. The source code of the Licensed Software must remain inside your official premises, as you have informed us. You will lock the source code of the Licensed Software and all copies thereof in a secured storage inside your official premises at all times when the source code of the Licensed Software is not being used as permitted under this Agreement. You will inform all Your employees who are given access to the source code of the Licensed Software of the foregoing requirements, and You will take all reasonable precautions to ensure and monitor their compliance with such requirements. You agree to promptly notify ST in the event of a violation of any of the foregoing, and to cooperate with ST to take any remedial action appropriate to address the violation. You shall keep accurate records with respect to its use of the source code of the Licensed Software. In the event ST demonstrates to You a reasonable belief that the source code of the Licensed Software has been used or distributed in violation of this Agreement, ST may by written notification request certification as to whether such unauthorized use or distribution has occurred. You shall reasonably cooperate and assist ST in its determination of whether there has been unauthorized use or distribution of the source code of the Licensed Software and will take appropriate steps to remedy any unauthorized use or distribution. You agree that ST shall have the right (where ST reasonably suspects that the terms and conditions of this Agreement with reference to Restriction clause have not been complied with) upon reasonable notice to enter Your's official premises in order to verify your compliance with this Restriction clause.

9.5 No warranty

The Licensed Software is provided "as is" and "with all faults" without warranty of any kind expressed or implied. ST and its licensors expressly disclaim all warranties, expressed, implied or otherwise, including without limitation, warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights. ST does not warrant that the use in whole or in part of the Licensed Software will be interrupted or error free, will meet your requirements, or will operate with the combination of hardware and software selected by You.

You are responsible for determining whether the Licensed Software will be suitable for your intended use or application or will achieve your intended results.

ST has not authorized anyone to make any representation or warranty for the Licensed Software, and any technical, applications or design information or advice, quality characterization, reliability data or other services provided by ST shall not constitute any representation or warranty by ST or alter this disclaimer or warranty, and in no additional obligations or liabilities shall arise from ST's providing such information or services. ST does not assume or authorize any other person to assume for it any other liability in connection with its Licensed Software.

Nothing contained in this Agreement will be construed as : (i) a warranty or representation by ST to maintain production of any ST device or other hardware or software with which the Licensed Software may be used or to otherwise maintain or support the Licensed Software in any manner; and (ii) a commitment from ST and/or its licensors to bring or prosecute actions or suits against third parties for infringement of any of the rights licensed hereby, or conferring any rights to bring or prosecute actions or suits against third parties for infringement. However, ST has the right to terminate this Agreement immediately upon receiving notice of any claim, suit or proceeding that alleges that the Licensed Software or your use or distribution of the Licensed Software infringes any third party intellectual property rights. All other warranties, conditions or other terms implied by law are excluded to the fullest extent permitted by law.

9.6 Limitation of liabilities

In no event ST or its licensors shall be liable to You or any third party for any indirect, special, consequential, incidental, punitive damages or other damages (including but not limited to, the cost of labour, re-qualification, delay, loss of profits, loss of revenues, loss of data, costs of procurement of substitute goods or services or the like) whether based on contract, tort, or any other legal theory, relating to or in connection with the Licensed Software, the documentation or this Agreement, even if ST has been advised of the possibility of such damages.

In no event shall ST's liability to You or any third party under this Agreement, including any claim with respect of any third party intellectual property rights, for any cause of action exceed 100 US$. This section does not apply to the extent prohibited by law. For the purposes of this section, any liability of ST shall be treated in the aggregate.

9.7 Termination

ST may terminate this license at any time if You are in breach of any of its terms and conditions. Upon termination, You will immediately destroy or return all copies of the software and documentation to ST.

9.8 Applicable law and jurisdiction

In case of dispute and in the absence of an amicable settlement, the only competent jurisdiction shall be the Courts of Geneva, Switzerland. The applicable law shall be the law of Switzerland.

9.9 Severability

If any provision of this agreement is or becomes, at any time or for any reason, unenforceable or invalid, no other provision of this agreement shall be affected thereby, and the remaining provisions of this agreement shall continue with the same force and effect as if such unenforceable or invalid provisions had not been inserted in this Agreement.

9.10 Waiver

The waiver by either party of any breach of any provisions of this Agreement shall not operate or be construed as a waiver of any other or a subsequent breach of the same or a different provision.

9.11 Relationship of the parties

Nothing in this Agreement shall create, or be deemed to create, a partnership or the relationship of principal and agent or employer and employee between the Parties. Neither Party has the authority or power to bind, to contract in the name of or to create a liability for the other in any way or for any purpose.

Revision history

Table 4. Document revision history

Date	Revision	Changes
17-Apr-2012	1	Initial release.
27-Sep-2012	2	Changed version references to LSP 3.2.5 and updated product references of evaluation boards throughout document.