HOW-TO GUIDE: USING SOURCERY CODEBENCH AND MENTOR EMBEDDED SOURCERY PROBE TO DEBUG THE LINUX KERNEL

KATHLEEN OLIVER, TECHNICAL MARKETING ENGINEER

'HOW-TO' WHITEPAPER

Mentor Graphics®

EMBEDDED SOFTWARE

www.mentor.com

ABSTRACT

This How-to Guide provides a detailed, step-by-step introduction to building and debugging the Linux® kernel with Sourcery™ CodeBench Professional and a Mentor Embedded Sourcery probe device - using a PandaBoard development board equipped with an OMAP 4430 processor as the target hardware. This paper demonstrates how to use features of the Sourcery CodeBench IDE to simplify the processes of configuring the target board, building the kernel, navigating around the kernel sources, and controlling the Sourcery probe debugger.

INTRODUCTION

Linux is becoming popular on embedded platforms due to many factors, including its openness, availability on many systems, and robustness. In some cases, taking full advantage of the target hardware may require modification of the Linux kernel or development of loadable kernel modules. Becoming familiar with and debugging such a large code base can be daunting to an engineer new to Linux kernel development. This paper shows how you can use Sourcery CodeBench to ease the initial board bring-up, explore the kernel source code, and debug a running kernel. Most steps can be accomplished directly from the Sourcery CodeBench IDE.

This paper is divided into sections that cover the various stages in bringing up a PandaBoard board, using Sourcery CodeBench to build the kernel, and finally, using a Mentor Embedded Sourcery Probe to debug the running kernel.

- Section 2, "Requirement and Environmental Setup" covers what you need need to know and what you have to complete the steps in this paper.
- Section 3, "Building the Linux Kernel" covers configuring the kernel and using Sourcery CodeBench to build it.
- Section 4, "Preparing the Target" covers connecting the board and making it ready for development. This section introduces the Sourcery CodeBench IDE's built-in terminal for serial I/O and the Remote Systems Explorer for browsing files on the target.
- Section 5, "Debugging" covers configuring the debug connection, attaching to the running kernel, and performing the common debug tasks.

This paper focuses on the initial steps required to get started with kernel debugging. Another Mentor Graphics How-to Guide, *Using Sourcery CodeBench to Develop and Debug a Linux Kernel Module,* details the steps required to build and debug a loadable kernel module using the same basic setup described in this paper.

REQUIREMENTS AND ENVIRONMENTAL SETUP

This paper assumes that you have a basic knowledge of Linux systems, C programming, embedded software, the Linux kernel build system, and Sourcery CodeBench. A Linux host with a properly installed cross-compiler is required to build the kernel. The procedures detailed in this document were tested using the following environment:

- The host for building and debugging the kernel is Ubuntu 10.04 32-bit.
- The `uboot-mkimage` and `lzop` packages are installed.

- Sourcery CodeBench Professional for ARM GNU/Linux 2011.09 or newer installed on the host system.
  TIP: You can download a 30-day evaluation of Sourcery CodeBench for ARM GNU/Linux from: http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/evaluations/arm-gnulinux
- The target is a PandaBoard Development Kit equipped with an OMAP 4430 processor.
- The board bootloaders, kernel, root filesystem and source code for the PandaBoard come from the Mentor Embedded Linux Kit for PandaBoard available from: http://go.mentor.com/linux-kits/
- The kernel is debugged over JTAG using Mentor Embedded Sourcery Probe (MESP) Personal for ARM via the Sourcery CodeBench Debug Sprite. Note that these steps also work for MESP Professional.
- In order to debug a PandaBoard using MESP and Sourcery CodeBench for ARM/GNU Linux 2011.09, you must download and install a few extra files. To do this, follow the instructions at: https://sourcery.mentor.com/GNUToolchain/kbentry255?@ok_message=kbentry%20255%20customers%20edited%20ok&@template=item

BUILDING AND BOOTING THE LINUX KERNEL

The Linux kernel features a huge collection of configuration options. Unfortunately, this means that a pre-built kernel, such as the one that comes with the board, often doesn't include all the features that you will need. This section covers configuring and building a kernel better suited to kernel debugging, and shows how to replace the stock image with the new image.

While the PandaBoard website publishes system images with a pre-built Linux kernel, for development purposes, we must build a kernel from source that includes debugging information. We'll be using the Mentor Embedded Linux Kit for Panda for the bootloaders, initial kernel, and root filesystem.

PREREQUISITES

- Visit http://go.mentor.com/linux-kits/ to download both the `mek-kit-pandaboard.bin` installer, as well as the `mek-kit-pandaboard_sources.tar` source file archive.

PREPARING THE LINUX KERNEL

The first step is to fetch the kernel source. The following steps are all performed on your Linux host. We'll next touch the board when it is time to load the new kernel on to it.

1. Unpack the source code from the `.tar` archive into `~/mel-kit-pandaboard_sources:`

```
$ tar -C ~/mel-kit-pandaboard_sources -xf <DOWNLOAD DIRECTORY>/
mek-kit-pandaboard_sources.tar
```

2. CD into the kernel sources directory:

```
$ cd ~/mel-kit-pandaboard_sources/copyleft_sources/linux-omap4-2.6.35.7-r0c
```

3. Check out the kernel release:

```
$ git clone  dev.omapzoom.org.pub.scm.integration.kernel-ubuntu.git
panda_kernel
```

4. This creates a directory called `panda _ kernel`. CD to that directory and switch the working branch to `glp1.4`:

```
$ cd panda _ kernel
$ git checkout glp1.4
```

5. Patch the kernel. The step below applies all the patches that are included in the MEL Kit source archive:

```
$ for x in `ls -1 ../*.patch`; do patch -p1 < $x; done;
```

6. By default, the Linux kernel is configured to build for the host that it is running on. To build for a given architecture, your build command line should look like this:

```
make ARCH=arm CROSS _ COMPILE=arm-none-linux-gnueabi- <target>
```

TIP: The `CROSS _ COMPILE` variable sets the prefix for the compiler, linker and any other commands used to build for the target. <target> is the proper build target for your board, which might be `zImage` or `uImage`, etc.

7. Make sure that the Sourcery CodeBench compiler is in your path:

```
arm-none-linux-gnueabi-gcc --version
```

This should return:

```
arm-none-linux-gnueabi-gcc (Sourcery CodeBench 2011.09-64) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

NOTE: The version information in the output shown above may vary depending on your compiler version.

The build environment is now set up. The next step is to configure the kernel for this board and to turn on the extra debugging features.

8. Configure the kernel using the MEL Kit for PandaBoard configuration:

a. Copy the kernel configuration file for the MEL Kit for PandaBoard into your kernel source tree:

```
$ cp ~/mel-kit-pandaboard _ sources/copyleft _ sources/linux-omap4-2.6.35.7-
r0c/defconfig
~/mel-kit-pandaboard _ sources/copyleft _ sources/linux-omap4-2.6.35.7-r0c/
panda _ kernel/.config
```

b. Run `oldconfig` to configure the Linux kernel source tree for pandaboard:

```
make ARCH=arm CROSS _ COMIPLE=arm-none-linux-gnueabi- oldconfig
```

9. Bring up the kernel configuration system:

```
make ARCH=arm CROSS _ COMIPLE=arm-none-linux-gnueabi- menuconfig
```

a. Go to the `Kernel Features` section, select `Use the ARM EABI to compile the kernel,` (ie make sure it is enabled with an asterisk beside the feature) and select `Exit.`

b. Go to the `Kernel hacking` section.

i. Enable the `Debug Filesystem.`

ii. Enable `Kernel debugging.` The screen will change and a number of new, debug-related options will come visible.

iii. Enable `Compile the kernel with debug info.` This turns on the standard `-g` option to the compiler and linker and expands the `vmlinux` image to include all the symbolic, line-level debugging information needed later.

c. Select `Exit, Exit,` and select `Yes` to save the new configuration.

NOTE: The above selections may already be enabled in your default kernel configuration

RESULTS

Your kernel is now configured and you are ready to import the source in to the Sourcery CodeBench IDE and build it.

BUILDING THE KERNEL

Now that the kernel has been configured, start Sourcery CodeBench IDE and create the workspace that will hold the project and debug settings. A workspace is the IDE's way of grouping together related projects and settings. Having different workspaces for different tasks allows you to easily switch between jobs and quickly pick up where you last left off.

This step imports the kernel in to the Sourcery CodeBench IDE and builds it. The result of this will be an image that you can deploy to the board.

1. Start the Sourcery CodeBench IDE:

```
./<INSTALLATION _ PATH>/CodeSourcery/Sourcery _ CodeBench _ for _ ARM _ GNU _ Linux/
bin/sourcerygxx-ide
```

NOTE: During installation of Sourcery CodeBench Professional edition, shortcuts to the executable were installed in a directory supplied during the installation. You can use your file browser to browse to this shortcuts directory and simply double-click on the icon representing Sourcery CodeBench IDE.

2. When the Workspace Launcher opens, enter a new Workspace and click **OK**:

```
/home/USERNAME/workspace _ panda
```

NOTE: You may wish to use your current workspace for this exercise, there is no requirement to select a new workspace unless you wish to.

3. If the Welcome screen appears, to close it, click **X** next to Welcome.

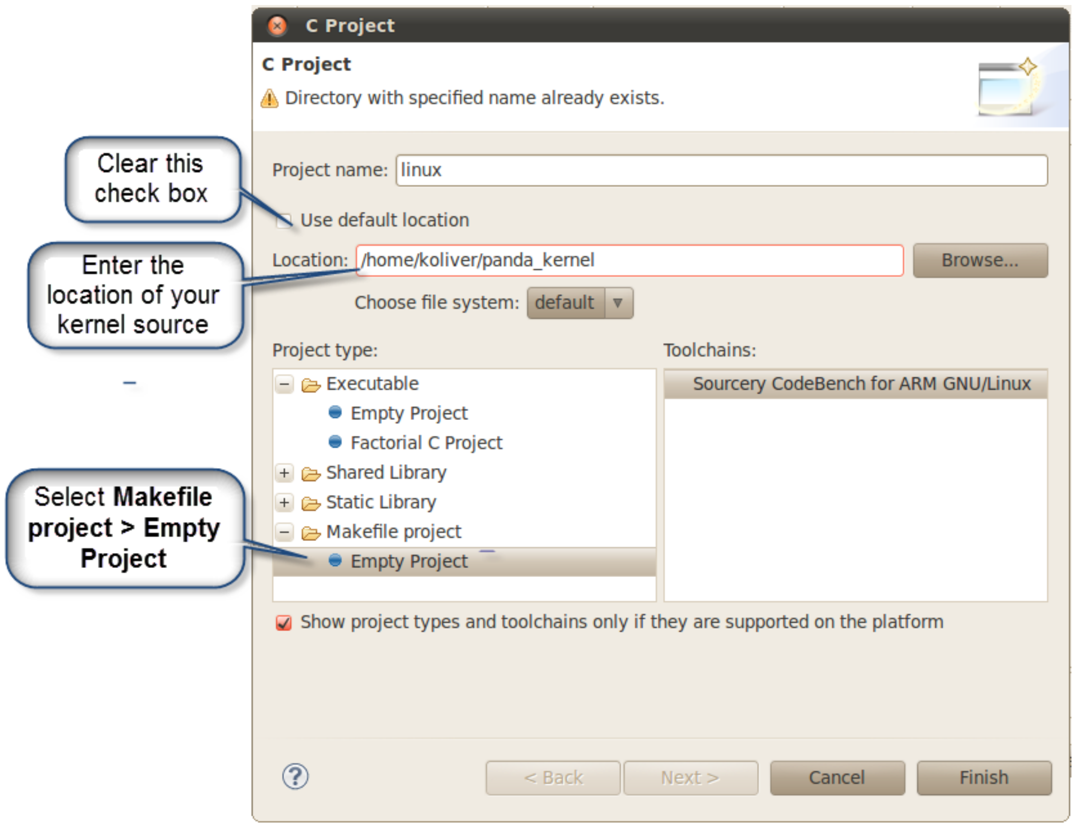
4. To create a new project, select `File → New → C Project`. This opens the C Project wizard.

   a. Enter *`linux`* for the project name.

   b. Clear `Use default location`.

   c. Browse to `~/mel-kit-pandaboard_sources/copyleft_sources/linux-omap4-2.6.35.7-r0c/panda_kernel` (the location of your top level kernel source tree.)

   d. Expand `makefile project` and select `Empty project.`


C Project
Clear this check box
Enter the location of your kernel source
Select Makefile project > Empty Project
Browse...
< Back
Next >
Cancel
Finish

C Project

Directory with specified name already exists.

Project name: linux

Use default location

Location: /home/koliver/panda_kernel

Choose file system: default

Project type:
- Executable
  - Empty Project
  - Factorial C Project
- Shared Library
- Static Library
- Makefile project
  - Empty Project

Toolchains:
- Sourcery CodeBench for ARM GNU/Linux

Show project types and toolchains only if they are supported on the platform

   e. Click `Finish.`

NOTE: The Sourcery CodeBench indexer starts up automatically and begins indexing the source files in the Linux source tree. This is indicated by the status frame at the bottom right of the IDE window. This may take quite some time depending on your host system. To view the status of the this, click **Show background operations.**

5. Check that `mkimage` and `lzop` are installed on your system.

6. Change the default make target in Sourcery CodeBench so it generates `uImage`:

   a. Right click on `linux` in the Project explorer pane in the C/C++ perspective and select **Properties**. The Properties dialog box for the `linux` project opens.

   b. Select C/C++ Build and select the **Builder Settings** tab.

   c. In Builder Settings, clear the **Use default build command** check box.

   d. Enter the proper make command in the box:

```
make ARCH=arm CROSS _ COMPILE=arm-none-linux-gnueabi- KCFLAGS=-mno-
unaligned-access uImage
```

NOTE: If want to speed up the build by using the parallel build system, add `-j n` to your `make` command to speed up the builds:

```
make -j 4 ARCH=arm CROSS _ COMPILE=...
```

NOTE: The `mno-unaligned-access-` flag prevents the compiler from emitting unaligned accesses. If you do not pass this option, your kernel will build, but it will not boot.

   e. Click **Apply**.

   f. Click **OK** to apply your changes and close the Properties dialog box.

7. To build your project, right-click your project and select **Build project**.

RESULTS

Once the build has finished the new Linux kernel will be available in *`arch/arm/boot/uImage`*.

PREPARING THE TARGET

Sourcery CodeBench includes many of the tools needed in day-to-day embedded development. In this section we will use the serial terminal to explore and configure the target.

The following setup is assumed for the rest of these instructions. It's worthwhile writing your setup down now for later reference.

- The IP address of MESP Personal is `169.254.156.162`
- The serial port is a FTDI USB-to-serial adapter that appears at `/dev/ttyUSB0` on the Linux host.
- All files including the project files and Linux kernel source code are assumed to be stored on the Linux host under your home directory at `/home/<username>/mel-kit-pandaboard _ sources/copyleft _ sources/linux-omap4-2.6.35.7-r0c/panda _ kernel`

BOOTING THE TARGET

1. Download and install the free Mentor Embedded Linux Kit for PandaBoard from http://www.mentor.com/embedded-software/downloads/linux-kits/.

2. Follow the instructions in the PandaBoard Installation and Getting Started Guide to configure your SD card.

3. Copy the kernel image (`uImage`) you just built to the boot partition of the SD card for the PandaBoard.

4. Connect RS232 port on the PandaBoard to the host over a straight-through serial cable.

5. Connect the ethernet cable to the same network as the host.

6. Connect the MESP Personal to the target and the host and follow the instructions in the Mentor Embedded Sourcery Probe User's Manual to configure it.

7. Power on the board.

RESULTS

The green **D1 LED** should be on. You are now ready to communicate with the target from the host.

USING THE SERIAL TERMINAL TO COMMUNICATE WITH THE TARGET

Accessing low-level functions such as the boot loader and initial console can only be done over the serial port. In this section we'll use the serial terminal included in the Sourcery CodeBench IDE to join the board to the network. Note that the terminal is included in every Sourcery CodeBench platform, so you do not need to install a separate terminal emulator utility. The serial terminal is available as a view which can be opened up from any perspective

WHAT ARE PERSPECTIVES AND VIEWS?

Think of a perspective as looking onto the projects from a certain angle, such as how the debug perspective looks on to the project with a focus on the tools that are useful when debugging. A view is a window or tool that is visible in that perspective.

- To open the terminal view, select `Window > Show View > Other...`, select `Terminal,` and double-click `New Terminal Connection`.

Take some time to browse through the other views that are available and then open the terminal view.

TIP: Double-click on the title bar of a view to maximize it.

Now that the terminal is running, press the **reset** button on the board. The stage 1 boot, U-Boot, and the kernel startup will scroll past and finish with a command prompt.

The next step is to identify the ipaddress of the target or to assign an ipaddress to the target.

1. Switch to the terminal view.

2. Log-in to the board with a login of `root.` There is not a password.

3. In the terminal window, type `ifconfig usb1 192.168.0.110` to assign this network address to the board.

4. Test the network by running `ping 192.168.0.110` on the board. You should see replies coming back from the host.

That completes the preparation. The board is now set up, on the network, easy to operate with, and contains a debug-ready kernel.

DEBUGGING USING MENTOR EMBEDDED SOURCERY PROBE

Next you can use a Mentor Embedded Sourcery Probe (MESP) device controlled by the Sourcery CodeBench Debug Sprite to debug the Linux kernel running on the target.

The Sourcery CodeBench Debug Sprite is an application that acts as an intermediary between the low-level JTAG debug library and the higher-level remote debug interface used by the debugger. The Debug Sprite does not require drivers or other software to use the MESP device.

- The Sprite is normally launched automatically from the Sourcery CodeBench IDE.
- Ensure the MESP is connected to both the target and the host.

PROCEDURE

1. Select `linux` in the Project Explorer.

2. To configure your debug session, select `Run → Debug Configurations....` This opens `the Debug Configurations dialog` box.

3. To create a new debug configurations. double-click on `Sourcery CodeBench Kernel Debug`. This creates a new configuration called `linux` Debug. Note that if you don't select the `linux` project first, the debug configuration will not be correctly associated with the `linux` project.

4. On the Main tab of the debug configuration, to set the kernel as the application to be debugged:

   a. Click **Search Project...** next to C/C++ Application.

   b. Select **vmlinux**, select **armle - /linux/vmlinux** as the Qualifier, and click **OK**.

   c. Click **Apply**.

5. To choose and configure your debug interface, select the Debugger tab:

   a. From the Debug Interface pull-down menu, select **Sourcery Probe**.

   b. From the Device pull-down menu, select your MESP device (for example, MESP-Personal / ARM ...).

Note: If you do not see a MESP device in the list, select **Manually Configured Probe** and enter the ipaddress for your probe.

c. From the Board pull-down menu, select **pandaboard**.

d. In the Debugger Options area:

i. Verify that the Target init script is set to `pandaboard.maj` and the CPU ID is **`cortex-a9`**. These options should be set automatically when you set the Board to pandaboard.

ii. Select **Core** and set it to **1**.

iii. Click **Apply**.

6. The debug configuration is now set up. To start your debug session, click **Debug**.

a. When the Confirm Perspective Switch dialog comes up, select **Remember my decision** and click **Yes** to switch to the debug perspective.

7. Starting the session will cause the board halt. Click **Resume** to resume executing.

The system is now connected up and ready to debug.

- Try clicking the **Pause** button. You should see the backtrace come up in the Debug view.
- Try double-clicking on different lines in the backtrace to see how the IDE automatically jumps to the right file and line. Note how the local variables are shown in the Variables view to the right.
- Click on the **Resume** button to let the kernel continue.
- To stop debugging, press the **Stop** button. Note that this will halt the target. In some situations Disconnect is better as it may leave the board running.
- To switch back to the **C++ perspective** you started in, click on the C/C++ Perspective button. Sourcery CodeBench has remembered your debug configuration and automatically made it the default.
- Click on the **Debug** button to start debugging again and automatically switch to the Debug perspective.

HINTS

This section contains some hints that may help you while using Sourcery CodeBench with this board.

- Pressing reset on the board will break the JTAG connection and cause a Target selection failed message in the IDE. The EmbeddedICE-RT that is built into the CPU is reset just like any other peripheral, breaking the connection with the Sourcery CodeBench Debug Sprite. The solution is to restart the Sourcery CodeBench Debug Sprite and start again.
- Make sure you are using a power-safe file system on the board, such as the default JFFS2 or alternatives such as YAFFS or UBIFS. Kernel debugging is quite intrusive and is more likely to cause corruption in file systems that are not power safe.
- If you see Ignoring packet error, continuing..., try restarting the Sourcery CodeBench Debug Sprite.

- If the debugger can't find the source for the current location, then make sure the kernel configuration has Compile the kernel with debug info and rebuild. If the debugger seems to stop at a strange location when you click Pause, then make sure the correct kernel is flashed into the board.

NEXT STEPS

You now have an environment set up for building and debugging the Linux kernel on your development board. Sourcery CodeBench Application Note AN003, *Using Sourcery CodeBench to Develop and Debug a Linux Kernel Module,* picks up after this note and takes you through the development process of an example kernel module. Similar to this note, it shows how to build and debug but also covers the special concerns and problems when developing kernel modules.

CONCLUSION

This paper has covered using Sourcery CodeBench to ease the Linux kernel development process. We've set up early serial communication, bootstrapped the network, configured a debug-able board-specific image, explored the Linux code base, and remotely debugged the kernel on a development board. These tasks, covering the range from early bring-up to day-to-day debugging, were done almost entirely within the Sourcery CodeBench IDE.

FURTHER READINGS

1. Corbet, Rubini, Kroah-Hartman (2005) "Linux Device Drivers, Third Edition" O'Reilly Media, Inc. Available in print or online.

2. Chapter 2 "Building and Running Modules" and chapter 4 "Debugging Techniques" are particularly useful, chapter 6 http://lwn.net/Kernel/LDD3/

3. Refer to the Getting Started guide included with your Sourcery CodeBench installation for a tutorial introduction to the Sourcery CodeBench IDE and more information about using the Sourcery CodeBench Debug Sprite.

For additional information, please visit: mentor.com/embedded

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
Room 1001, 10F
International Trade Building
No. 333, Section 1, Keelung Road
Taipei, Taiwan, ROC
Phone: 886.2.87252000
Fax: 886.2.27576027

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor Graphics®

MGC 05-12 TECH10620-w