HOW-TO GUIDE: BUILDING A LINUX KERNEL USING THE SOURCERY CODEBENCH IDE

RICARDO ANGUIANO, TECHNICAL MARKETING ENGINEER

'HOW-TO' WHITEPAPER

Mentor Graphics®

EMBEDDED SOFTWARE

www.mentor.com

INTRODUCTION

This document demonstrates various techniques using the Mentor® Embedded Sourcery™ CodeBench IDE to build a Linux® kernel and kernel modules. This paper also demonstrates how to use Sourcery CodeBench to assist in board bring-up and navigate around a large, complex project such as the Linux kernel. The emphasis is on maximizing the time spent in a friendly desktop UI, making this an ideal environment for developers who are more comfortable using an IDE rather than low-level, command-line tools and kernel build procedures.

This paper assumes that you have a basic conceptual knowledge of embedded systems. To keep things brief many of the intermediate steps are omitted.

REQUIREMENTS

- The host for running Sourcery CodeBench is an x86 computer running the 32-bit Ubuntu Linux 10.04 operating system.
- The `uboot-mkimage` and `lzop` packages are installed on the host system.
- Sourcery CodeBench Professional for ARM GNU/Linux 2011.09 or newer installed on the host system.
- A memory card reader capable of reading and writing to secure digital (SD) compatible memory cards.
- The target system is a PandaBoard Development Kit equipped with an OMAP 4430 processor.
- The bootloaders, kernel, root filesystem and source code for the PandaBoard come from the Mentor Embedded Linux (MEL) Kit for PandaBoard available from: http://go.mentor.com/linux-kits/

Sourcery CodeBench supports a wide range of hosts, JTAG units, and target processors. For more specific information, visit: http://go.mentor.com/codebench/

PREPARING THE KERNEL SOURCES

You need to prepare the Linux kernel sources to make sure that you are building the right version and have applied the right patches. While the PandaBoard website publishes system images with a pre-built Linux kernel, for development purposes you must build a kernel from source that includes debugging information. The Mentor Embedded Linux Kit for Panda provides precompiled bootloaders, a precompiled kernel, a root file system, and source code.

PREREQUISITES

- Visit: http://go.mentor.com/linux-kits/ to download both the `.bin` installer, as well as the `.tar` source file archive.
- Follow instructions in the PandaBoard Installation and Quick Start Guide to setup the board.
- Unpack the source code from the `.tar` archive into `~/mel-kit-pandaboard _ sources.`

PROCEDURE

1. Use the command line to go to the directory containing the Linux kernel source code from the MEL Kit for Panda:

```
$ cd ~/mel-kit-pandaboard _ sources/copyleft _ sources/linux-omap4-2.6.35.7-r0c
```

2. Create a working tree directory named `src` using the `git clone` command on the repository directory:

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

3. Go to the `src` library:

```
$ cd ~/mel-kit-pandaboard _ sources/copyleft _ sources/linux-omap4-2.6.35.7-r0c/src
```

4. Check out the `glp1.4` branch:

```
$ git checkout glp1.4
```

5. Apply the kernel patches included in the MEL Kit source archive:

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

6. Copy the default Kernel configuration included in the MEL Kit source archive:

```
$ cp ../defconfig .config
```

7. Verify the kernel version:

```
$ head -4 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 35
EXTRAVERSION = .7
```

The output indicates that you have 2.6.35.7. kernel source tree.

RESULTS

You have now checked out the correct source code branch, applied patches included in the MEL Kit for PandaBoard, and copied the existing MEL Kit for PandaBoard kernel configuration file into your source tree.

PREPARING TO BUILD A KERNEL USING THE CODEBENCH IDE

Now you are ready to import your Linux kernel source code into the Sourcery CodeBench IDE, and configure your project.

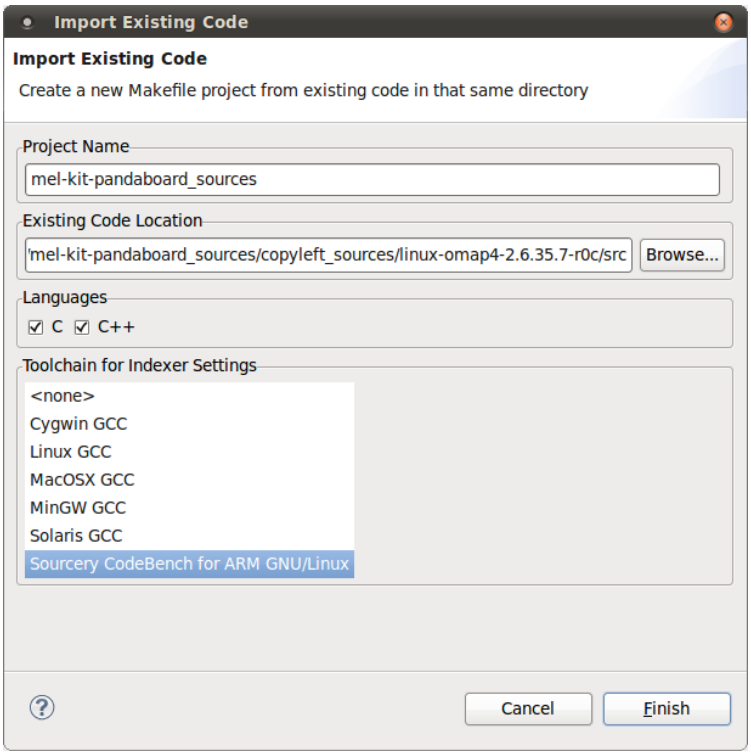
PROCEDURE

1. Start the Sourcery CodeBench IDE from the command line and create a new workspace to hold your work:

```
$ ~/CodeSourcery/Sourcery _ CodeBench _ for _ ARM _ GNU _ Linux/bin/sourcerygxx-ide
```

NOTE: The Sourcery CodeBench installer created a directory in your home directory with shortcuts to documentation and executables. The default name for this directory is Sourcery_CodeBench_for_ARM_EABI. You can use your file browser to browse the shortcuts directory and simply double-click on the icon representing Sourcery CodeBench IDE.

2. Create a new project.



a. Select **File > New > Makefile Project with Existing Code**.

b. Name your project `mel-kit-pandaboard _ sources`.

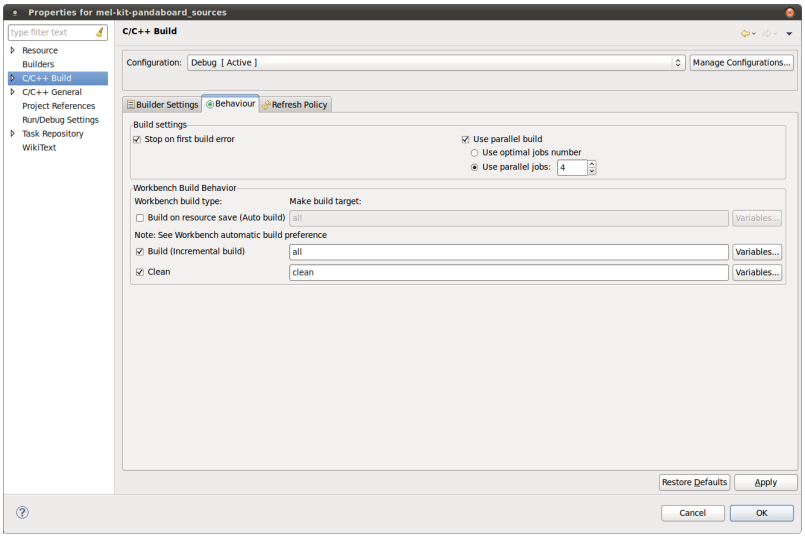
c. Browse to the `src` directory you prepared above for the Existing Code Location.

d. Select both the **C** and **C++** Language checkboxes.

e. Select **Sourcery CodeBench for ARM GNU/Linux** under Toolchain for Indexer Settings.

f. Click **Finish**.

3. Modify the build properties for this project.



a. Select **Projects > properties.**

b. Select **C/C++ Build.**

c. Select the **Behavior** tab.

d. Enable Parallel Builds.

i. Select the **Use Parallel Build** option.

ii. Select **Use Parallel Jobs**.

iii. Enter the number of parallel jobs to run.

NOTE: The kernel build scales very well so that a parallel build on a quad core machine will finish almost four times faster. Using the same number of parallel jobs as cores on your build host is a reasonable choice.

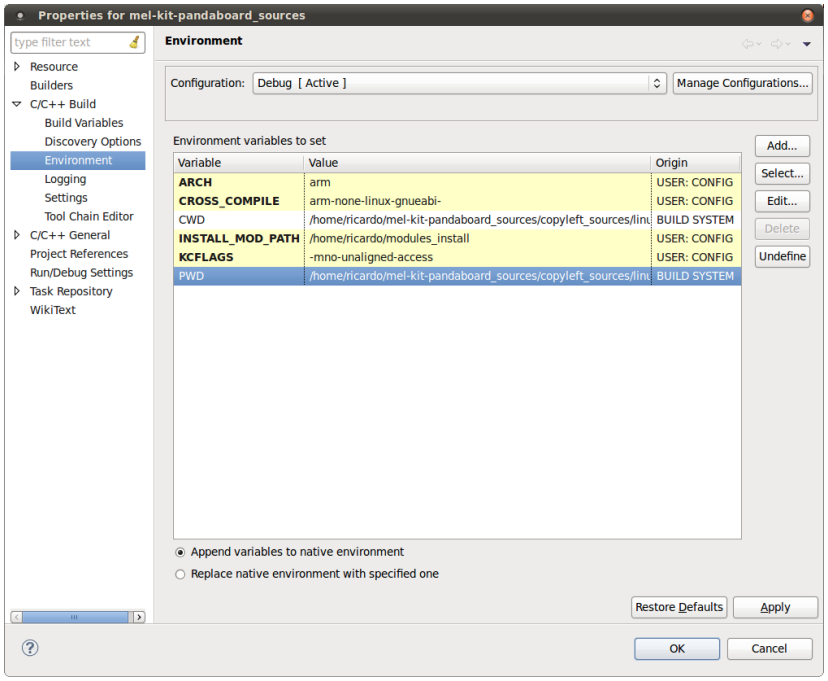
IMPORTANT: Do not select the **Use Optimal jobs number** option, as it spawns unlimited jobs which can cause your system to run out of memory and crash your machine.

e. Disable Automatic Builds.

i. Clear the **Build on Resource save (Auto build)** check box. You have more control over when you build the project if you disable automatic builds.

f. Click **OK** to save your changes.

4. Add environment variables to modify the kernel build process.



a. Select and expand **Project > Properties > C/C++ Build** and select **Environment.**

b. Click **Add...** on the right side of the dialog box to add the following environment variables.

Variable Name	Value	Meaning for Kernel build system
ARCH	arm	Define the target architecture
CROSS_COMPILE	arm-none-linux-gnueabi-	Define the cross compiler prefix
KCFLAGS	-mno-unaligned-access	Pass additional toolchain flags
INSTALL_MOD_PATH	~/modules_install	Location for kernel module install

The **ARCH** variable tells the Kernel build system that you are targeting the ARM architecture. The **CROSS_COMPILE** variable tells the Kernel build system the toolchain prefix for the cross compiler. The **KCFLAGS** variable passes additional toolchain options. In this case, the `-mno-unassigned-access` flag avoids an optimization that causes problems for kernel builds. The **INSTALL_MOD_PATH** variable indicates where the build system should install the kernel modules.

c. Click **OK** to save your changes.

RESULTS

Up to this point, you have now started the Sourcery CodeBench IDE, created a new project, and imported your Linux source code, configured the project settings, and setup environment variables in preparation for building your Linux kernel.

CREATING MAKEFILE TARGETS

To understand the rest of this build process, it is helpful to understand the traditional kernel build process. Normally, when building a kernel the following steps occur:

1. Configure the kernel to choose your kernel feature options (`make config`).

NOTE: There are other kernel makefile targets which will configure your kernel besides the `config` makefile target. The `menuconfig` makefile target will display a curses based text interface on the command line. The `xconfig` and `gconfig` makefile targets will display a GUI interface. All these makefile targets accomplish the same task, but the user interfaces are different.

2. Build the kernel and modules `(make all)`.

3. Install the kernel and modules `(make install)`.

A kernel config file was provided for you and you have already copied it into the top level source tree as part of the Preparing the Kernel Sources section above. When reusing an existing kernel configuration, you must use the `oldconfig` makefile target instead of the `config` makefile target. In addition to building the kernel, you want to use the `uImage` makefile target so that the kernel build system builds a uImage kernel file for use with the U-Boot bootloader used on PandaBoard. Lastly, you will not use the `install` makefile target. You are building a kernel and modules for the ARM architecture so installing them on an x86 host does not make sense. Instead, you will copy the files onto the media used to boot the PandaBoard.

Our build process will look like this:

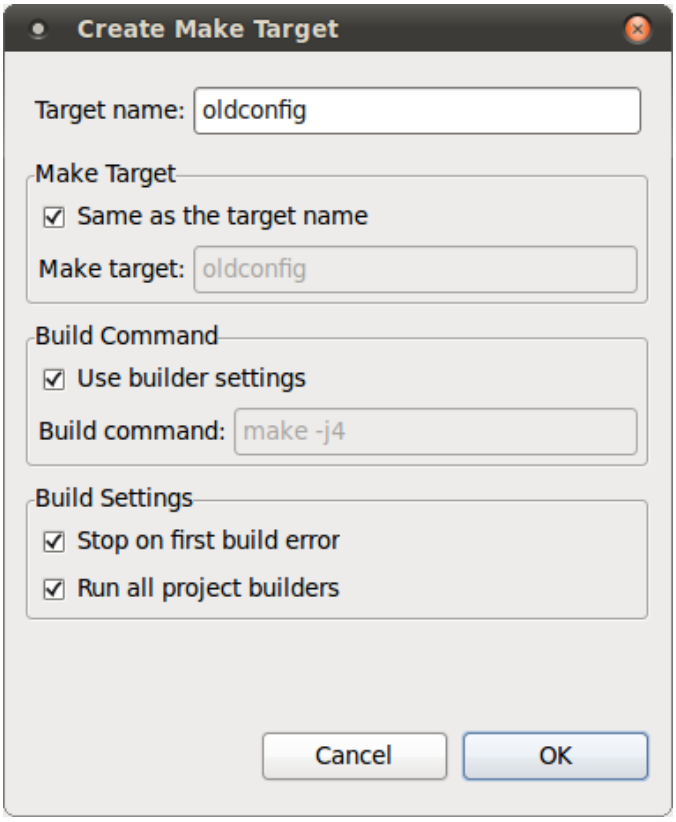
1. Configure the kernel with a preexisting configuration file `(make oldconfig)`.

2. Build the kernel and package it into the uImage file `(make uImage)`.

3. Build the modules `(make modules)`.

4. Install the modules to the module installation directory specified by the `INSTALL _ MOD _ PATH` environment variable. `(make modules _ install)`.

5. Copy the `uImage` file and modules directory to the PandaBoard's boot media.



PROCEDURE

1. Create makefile targets in Sourcery CodeBench IDE.

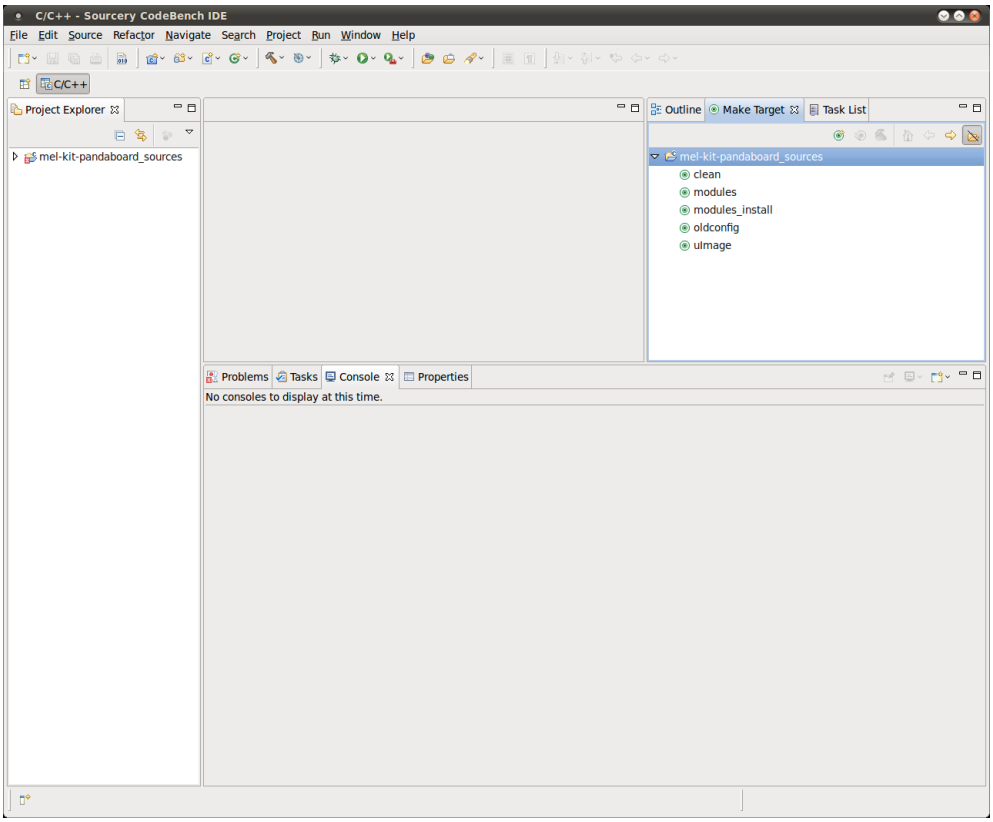
   a. Select **Project > Make Target > Create**

2. Create make targets for `clean`, `modules`, `modules_install`, `oldconfig`, and `uImage`.

3. Display the makefile targets in the C/C++ perspective.

   a. Select the **Make Target** tab with the green circle icon in the upper right side of the screen.

   b. Click on the **Hide Empty Folders** icon.



RESULTS

You have created the required makefile targets in the Sourcery CodeBench IDE to build the Linux kernel.

BUILDING A KERNEL AND MODULES

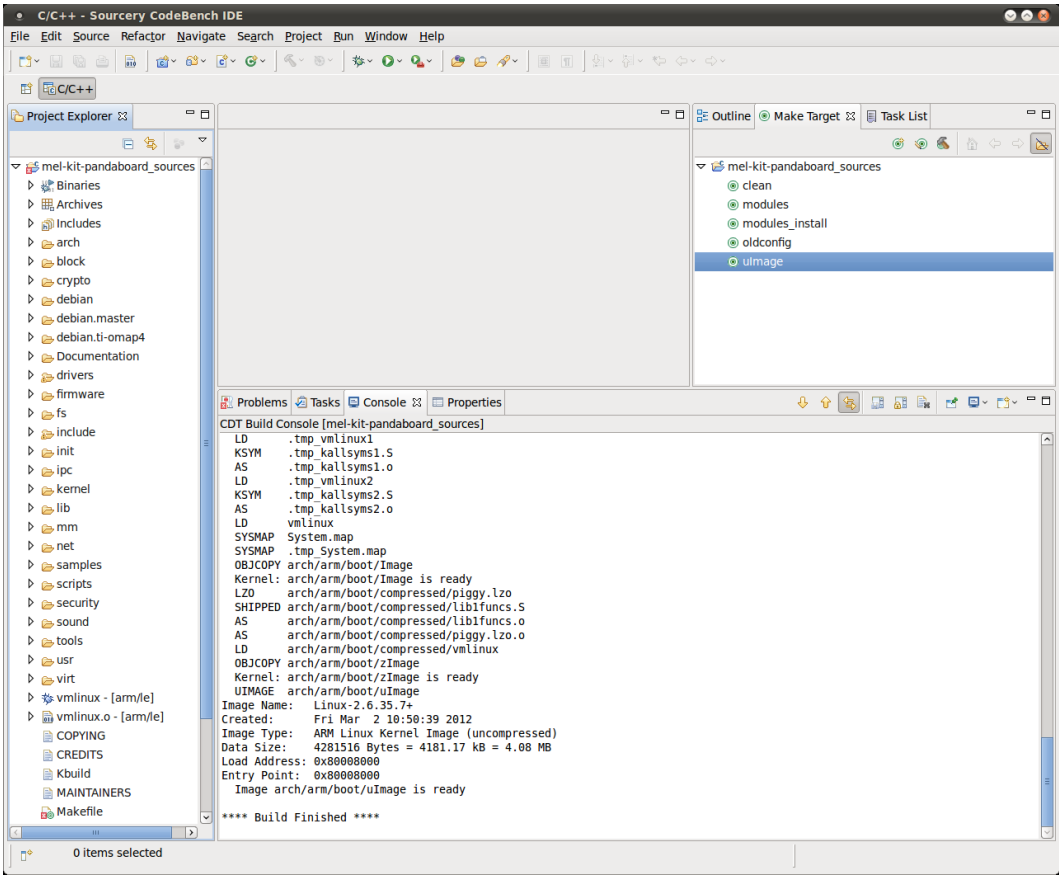
You are now ready to build the Linux kernel and modules.

PROCEDURE

1. Double click on the targets in the **Make Target** tab in the following order:

Make Target Invocation Order	Purpose
`clean`	Ensure that you are starting with a clean build
`oldconfig`	Configure the source tree based on the `.config` file already in place
`uImage`	Build the kernel and package into a `uImage` file as required by U-Boot
`modules`	Build the kernel modules
`modules_install`	Install the kernel modules in the `INSTALL_MOD_PATH` directory

Here's a sample screenshot after the `uImage` build has been completed:



RESULTS

You have now built the Linux kernel and modules.

INSTALLING THE KERNEL AND MODULES

Follow these steps to install your newly built kernel and modules onto your PandaBoard boot media.

PROCEDURE

1. Safely shutdown and power off your PandaBoard.

2. Remove the memory card from your PandaBoard.

3. Insert the card into your memory card reader.

4. Connect the memory card reader to your build host where your kernel and modules reside. You will need `root` or `sudo` access on your host for all commands listed on the following page.

5. Mount the partitions from the memory card on the host:

```
$ sudo mount /dev/sdd1 /mnt/sdd1
$ sudo mount /dev/sdd2 /mnt/sdd2
```

6. Make a backup of your old kernel:

```
$ sudo cp /mnt/sdd1/uImage /mnt/sdd1/uImage.old
```

7. Copy your new kernel `uImage` file to the boot media:

```
$ sudo cp ~/mel-kit-pandaboard_sources/copyleft_sources/
linux-omap4-2.6.35.7-r0c/src/arch/arm/boot/uImage /mnt/sdd1
```

8. Make a backup of your old kernel modules:

```
$ sudo mv /mnt/sdd2/lib/modules/2.6.35.7+ /mnt/sdd2/lib/modules/2.6.35.7+.old
```

9. Copy your new kernel modules to the boot media:

```
$ sudo cp -a ~/modules_install/lib /mnt/sdd2
```

NOTE: Remember, the `INSTALL_MOD_PATH` environment variable defined the installation directory as `~/modules_install` in the section above titled ***Preparing to Build a Kernel using the CodeBench IDE***. When you built the `modules_install` makefile target above, the modules were installed in that directory.

10. Recursively change the ownership and group to `root` on the `lib` directory in the PandaBoard's root filesystem:

```
$ sudo chown -R root:root /mnt/sdd2/lib
```

11. Verify the contents of the PandaBoard's boot media's first partition with the `ls command`. The output should show MLO, u-boot.img, uImage and uImage.old files:

```
$ sudo ls /mnt/sdd1MLO  u-boot.img  uImage  uImage.old
```

12. Verify the contents of the PandaBoard's root filesystem /lib directory with the ls command. You should see directories with your new and old kernel modules:

```
$ sudo ls /mnt/sdd2/lib/modules
2.6.35.7+  2.6.35.7+.old
```

13. Unmount both partitions from the host:

```
$ sudo umount /mnt/sdd1
$ sudo umount /mnt/sdd2
```

14. Disconnect the memory card reader from the host.

15. Remove the memory card from the memory card reader.

16. Insert the memory card into your PandaBoard.

17. Power on the PandaBoard to boot your new kernel.

18. Login and verify that you booted your new kernel with the `dmesg` command:

```
# dmesg | grep "Linux Version"
Linux version 2.6.35.7+ (user@buildhost) (gcc version 4.6.1 (Sourcery CodeBench
2011.09-78) ) #3 SMP PREEMPT Thu Feb 23 08:43:28 PST 2012
```

RESULTS

You have now installed and booted your newly built Linux kernel and modules.

FURTHER READING

- Kernel build system documentation: http://kernel.org/doc/Documentation/kbuild/kbuild.txt
- Kernel makefile documentation: http://kernel.org/doc/Documentation/kbuild/makefiles.txt
- Kernel makefile target documentation: http://kernel.org/doc/makehelp.txt

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

MGC 05-12 1030520-w