Opengear Custom Development Kit (CDK)
User Guide

Introduction
~~~~~~~~~~~~~

This guide explains how to use the CDK to make custom firmware images. This includes building custom or user-defined binaries as well as customizing the Linux kernel and modules.

For more information visit:

http://opengear.com/faq284.html (CDK guide)
http://opengear.com/sourcecode.html (Installing from source code)

In this file:

System Requirements
~~~~~~~~~~~~~~~~~~~

These instructions are for a 32-bit x86 install of Ubuntu 10.04. KCS and CMS products will not build if you have a 64-bit distribution.

Different versions of the CDK are available from:

http://opengear.com/firmware/cdk/

The required packages for building Opengear firmware images can be downloaded from Synaptic Package Manager. The package names as found in Ubuntu's repositories are:

- zlib1g-dev
- bin86
- qemu
- genext2fs

The packages you need will depend on which type of product you are building for. KCS and CMS family products use an x86 processor. SD, IM, CM and ACM products use an ARM processor.

For building ARM based products (SD/IM/CM/ACM) you only need the first package.

```
$ sudo apt-get install zliblg-dev
```

For building x86 based products (KCS/CMS) you require all 4 packages.

```
$ sudo apt-get install zlib1g-dev bin86 qemu genext2fs
```

For the cross-compiling of the firmware for Opengear products it is required to install the necessary Linux tool chains. The KCS and CMS family products use the i386 tool chains while all other Opengear products use the ARM tool chains.

The tool chains can be downloaded from:

ftp://ftp.opengear.com/tools/

For KCS and CMS family products:

- i386-linux-tools-20030930.sh
- i386-linux-tools-20070808.sh

For all other Opengear products:

- arm-linux-tools-20061213.sh

Running these shell scripts as a super user will extract the tool chains to '/usr/local/opengear/' and place binaries into '/usr/local/bin/'. Once you have downloaded the shell scripts navigate to the download location and run the relevant script(s):

```
$ sudo ./arm-linux-tools-20061213.sh
```

Make Commands
~~~~~~~~~~~~~~

It is important to enter make commands from the root directory of the CDK as the root Makefile contains configuration settings. If you execute make commands from any other directory Linux will not use the correct cross-compiler.

Entering 'make help' gives the following information:

```
Quick reference for various supported make commands.
----------------------------------------------------------------
make config              configure the kernel/modules
make dep                 2.4 and earlier kernels need this step
```
~~~~~~~~~~~~~~

```
make                    build the entire tree and final images
make clean              clean out compiled files, but not config
make linux              compile the kernel only
make romfs              install all files to romfs directory
make image              combine romfs and kernel into final image
make modules            build all modules
make romfs.modules      install modules into romfs
make DIR_only           build just the directory DIR
make DIR_romfs          install files from directory DIR to romfs
make DIR_clean          clean just the directory DIR

Typically you want the following sequence:

$ make config           customize nothing for default config
$ make dep              (for 2.4.x kernel)
$ make linux
$ make modules
$ make
---------------------------------------------------------------------
```

An example of using the make DIR_* targets is if you already have a preconfigured and precompiled tree and wish to add a new application. You would not want to recompile everything, just your application; e.g. user/myapp. You may execute:

```
$ make myapp_only    (to compile your app only)
$ make myapp_romfs   (to install your app into the romfs)
$ make image         (combine kernel and romfs to build image)
```

Building An Image
~~~~~~~~~~~~~~~~~

Opengear images for ARM based devices are a read-only SQUASHFS-LZMA file system. As such, adding/removing applications involves generating a new file system.

In the Opengear build system the file system is generated from the romfs directory.

Building a custom image involves the following:

1. Compiling source (kernel, modules, applications)
2. Installing to romfs (modules->romfs, applications->romfs)
3. Building image (compressing romfs into a SQUASHFS-LZMA file system, and appending the kernel to the end of the image)

Please note that there are limitations for the size of the final image. For devices with 8MB of flash such as CM400x and SD400x the maximum image size is approximately 6MB. The default image is approximately 5.8MB if no applications are added. If you would like to add any applications you might consider removing unrequired features.
~~~~~~~~~~~~~~~~~

If you do not wish to make any customizations you can use the existing image or build your own using the precompiled kernel by typing 'make'. This will compile any new applications added under 'user/' but will not add them to the romfs. If you wish to add the applications to the romfs (so that they are included in the image) you will have to edit 'user/Makefile' to include a list of the applications you want to install to the romfs.

If you just want to build an image from a compiled kernel and built romfs simply execute 'make image'.

Customizing The Kernel
~~~~~~~~~~~~~~~~~~~~~~

You can modify the kernel configuration and application set generated for your target using the config system. You can configure by running:

```
$ make config
```

When you run 'make config' you are presented with a menu to customize the Linux kernel and any kernel modules you wish. You will then need to recompile the kernel and modules if you have made changes.

```
$ make config
$ make dep (if 2.4.x kernel)
$ make linux
$ make modules
```

The SD400x and CM400x product lines use a 2.4.x kernel, while the CM41xx, IM42xx, ACM500x, KCS and CMS products all use a 2.6.x kernel. If it is a 2.4.x kernel you will have to run 'make dep' first.

Adding Custom Binaries
~~~~~~~~~~~~~~~~~~~~~~

You can add custom binaries in the 'user/' directory. The CDK contains an example Hello World program, "hello.c", to show you how to compile your applications. Please refer to 'user/hello/Makefile' for an example Makefile. Note the romfs target. This is necessary to install your application into the romfs before building the final image. Please ensure your Makefiles contain romfs and clean targets.

In the romfs target use the $(ROMFSINST) command to copy files into the romfs. It takes two arguments; the source file and the destination location and filename within the romfs. For example:

```
$(ROMFSINST) build/bin/blah /bin/blah
```

will install build/bin/blah (path relative to makefile) to romfs/bin/blah.

Note: in order for the romfs targets in the user sub-directories to get executed you must edit 'user/Makefile' to add a list of the directories you want to include. Currently all the directories (hello, p7zip, smstools3, squashfs) get compiled but none of the built binaries are added to the romfs.

If you wish to add a standard GNU app for which you can download the tarball (*.tar.gz) from a static URL you might consider using automake. As an example, refer to 'user/smstools3/Makefile'. All that is necessary is the URL to the .tar.gz file of the GNU app and to include 'tools/automake.inc'.

For more details on automake please refer to 'tools/automake.txt'.

```
# example makefile with 4 lib dirs and 3 app dirs

dirs += \
   lib/aaa \
   lib/bbb \
   lib/ccc \
   lib/ddd \
   app/eee \
   app/fff \
   app/ggg

appdirs += \
   eee \
   fff \
   ggg

all:
   for i in $(dirs); do \
      if [ -d "$$i" ]; then \
         make -C $$i || exit $$? ; \
      fi; \
   done

romfs:
   for i in $(appdirs); do \
      if [ -f app/"$$i"/"$$i" ]; then \
         $(ROMFSINST) app/"$$i"/"$$i" /bin/"$$i" ; \
         fi; \
      done

clean:
   for i in $(dirs); do \
      if [ -d "$$i" ]; then \
         make -C $$i clean ; \
      fi;  \
   done
```

Installing The Image
~~~~~~~~~~~~~~~~~~~~

To install a new firmware image using the web UI click on Firmware from the System menu.

Note: your console server will not allow you to upgrade to the same or an earlier firmware version by default. To override this type '-i' into the Firmware Options field.

For more information refer to http://www.opengear.com/faq253.html.

Factory Reset
~~~~~~~~~~~~~

To perform a reset to factory settings push the erase button twice. A ball point pen or paperclip is a suitable tool for performing this procedure. Do not use a graphite pencil. Depress the button gently twice (within a couple of second period) while the unit is powered ON. This will reset the console server back to its factory default settings and clear the console server's stored configuration information.

The hard erase will clear all custom settings and return the unit back to factory default settings (i.e. the IP address will be reset to 192.168.0.1). You will be prompted to log in and must enter the default administration username and administration password:

```
Username: root
Password: default
```

System Recovery
~~~~~~~~~~~~~~~

Should the firmware become corrupt or unusable you can recover the system by performing a net-boot. Network booting is the process of booting from a network location (e.g. your development machine). This will load the image file from a specified network location into RAM using the primary ethernet port. The device will then boot from the image in RAM. After you have successfully performed a net-boot you will need to install a known working image using the web UI.

In order to net-boot from your development machine you will require a tftp server and a dhcp server configured appropriately.

```
$ sudo apt-get install atftpd dhcp3-server
```

These are example packages, you may use others if you wish.

Edit your dhcp server configuration (e.g. /etc/dhcpd.conf) to include an entry for the Opengear device. The entry might look something like:

```
  host myopengear
```
~~~~~~~~~~~~~~~~~~~~

```
{
  hardware ethernet 00:13:C6:00:00:01;
  fixed-address 192.168.0.1;
  filename "image.bin";
}
```

Important to note here is the "filename" parameter. By default when you build an image the generated file is called "image.bin" and is placed in the images folder. A copy is placed in '/tftpboot/' if it exists.

To affect a net-boot:

1. Turn the device off
2. Depress the erase button
3. Turn the device on while still holding the erase button

It is recommended to edit your tftp configuration to also use the '/tftpboot/' directory.

For a more detailed (Windows) guide please refer to:

http://www.opengear.com/faq263.html