CodeWarrior for Power Architecture

Compiling the Kernel and U-Boot Using System Builder

1 Using System Builder

You can follow the steps given below to compile the kernel and U-Boot using System Builder.

NOTE

The commands in this section are generic and can be customized for any host or target you choose. Please refer to the examples for target-specific commands. The examples in this application note are based on the P4080ds target.

1.1 Install system builder on your host machine

— As a root user, mount the ISO image on your machine.

```
mount -o loop <System Builder Image>.iso
<systembuilder_install_path>
```

— As a non-root user, install the system builder.

```
<systembuilder_install_path>/install
```

Contents

You will be prompted to input the appropriate System Builder install path. Ensure that you have the required permissions for the install path.

NOTE

There is no uninstall script. To uninstall System Builder you need to remove the *<systembuilder_install_path>/QorIQ-DPAA-SDK-<yyyymmdd>-systembuilder* directory manually.

1.2 Set up the host environment

To run System Builder, access the directory in which System Builder is installed and follow these steps to start using it.

a) Change the directory to where you installed system builder.

```
cd <system_builder_installation_path>
```

b) Run the preparation script.

```
./script/<linux_distribution>-oe.sh
```

Examples:

For Ubuntu8.04 — `./script/ubuntu804-oe.sh`

For Ubuntu9.10 and Ubuntu 10.04 — `./script/ubuntu-oe.sh`

For RHEL 5 and CentOS 5 — `./script/rhel5-oe.sh`

NOTE

You need to run the preparation script only once for each new installation and this action requires a sudo permission. This is done to ensure that the host packages that are required, such as a host compiler are installed and are of the appropriate version.

1.3 Create a Platform Development Kit (PDK) project

```
cd <system_builder_installation_path>
```

```
./scripts/create-config.py --config-file=fsl-<bsp>/sample-create-config<bit-type>.ini
-j 4 -t 2
```

Refer to Appendix B/-9 for more information on values for <*bsp*> for supported targets.

Example:

```
./scripts/create-config.py --config-file=fsl-p4080ds/sample-create-config.ini -j 4 -t 2
```

NOTE

You can use the -j and -t parameters to speed up build creation and control the amount of parallel processing that the build will use. The -j option determines the number of jobs to have the make program itself spawn during the compilation stage. Set -j equal to 1.5 * (number of CPU cores), rounded up to the nearest integer. The -t option specifies the maximum number of BitBake tasks (or threads) that can be issued in parallel. Set -t equal to the number of CPU CORES that you have. The default settings are -j 2 and -t 1. These options set the PARALLEL_MAKE (for -j) and BB_NUMBER_THREADS (for -t) variables in local.conf.

1.4 Set up cross-compile environment and build images

a) Change the directory to where you installed system builder.

```
cd <system_builder_installation_path>
```

b) Setup your shell environment by using the *bitbake.rc* file. The environment file must be loaded prior to building a project. Use the appropriate command for your shell to load a file into the environment.

```
source build_<bsp><bit-type>_release/bitbake.rc
```

Example:

```
source build_p4080ds_release/bitbake.rc
```

c) Generate images for uboot.bin, uImage, dtb files, FMAN UCODE and rootfs by using the following command. These images are stored in the *build_<bsp>_release/deploy/glibc/images/* folder.

```
bitbake standalone-environment-linux
```

NOTE

The *hv.uImage*, *hv*.dtb* and *rcw*.bin* files are also generated and stored in the *build_<bsp><bit-type>_release/deploy/glibc/images/boot/* folder.

1.5 Building U-Boot for debugging with the CodeWarrior Debugger

Follow the steps given below to build U-Boot for debugging with the CodeWarrior Debugger:

1. Change the directory to where you installed system builder.

```
cd <system_builder_installation_path>
```

2. Create a new custom collection, for example, `mycollection/`.

```
mkdir -p mycollection/recipes/u-boot
```

3. Change the directory to point to the new custom collection.

```
cd mycollection/recipes/u-boot
```

4. Create `amend.inc` file and add following content in the `amend.inc` file.

```
EXTRA_OEMAKE += "CONFIG_CW=1"
```

5. Update the project configuration.

```
./scripts/create-config.py --config-file=fsl-<bsp>/sample-create-config<bit-type>.ini
--override-collections mycollection -t 2 -j 4
```

Example:

```
./scripts/create-config.py --config-file=fsl-p4080ds/sample-create-config.ini
--override-collections mycollection -t 2 -j 4
```

d) Clean up the existing U-Boot work area and rebuild U-Boot.

```
bitbake -c clean u-boot
```

```
bitbake u-boot
```

The U-Boot binary image is placed in *build_<bsp><bit-type>_release/deploy/glibc/images/*.

NOTE

The binary images for kernel and U-Boot, the ramdisk, and the dtb files are placed in this location.

1.6 Building the Kernel for debugging with the CodeWarrior Debugger

Follow the steps given below to build kernel for debugging with the CodeWarrior Debugger:

1. Change the directory to where you installed system builder.

```
cd <system_builder_installation_path>
```

2. Install kernel source code (if required).

```
bitbake -c clean virtual/kernel
```

```
bitbake -c patch virtual/kernel
```

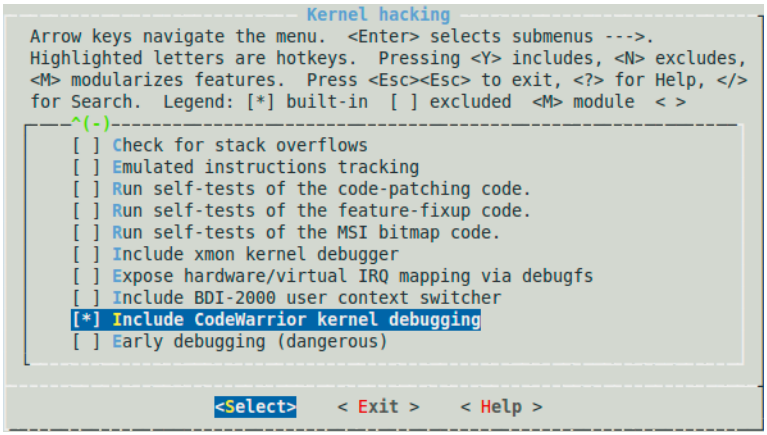
3. Configure kernel to enable CodeWarrior support.

```
bitbake -c menuconfig virtual/kernel
```

The kernel configuration user interface appears.

4. Scroll down the list and select **Enable kernel hacking**.
5. Select Include CodeWarrior kernel debugging by pressing Y. Select other desired configuration options for kernel debug.

Figure 1. Enabling CodeWarrior Kernel Debugging



6. Select **Exit**.
7. Rebuild kernel.

   ```
   bitbake virtual/kernel
   ```

   The vmlinux elf image is placed in
   build_<bsp><bit-type>_release/work/<bsp><bit-type>-linux/linux-3.0.6-r1/linux-3.0.6/

2 Configuring Target Hardware to Use the U-Boot Image

To use the target hardware to run and debug U-Boot, you need to burn the U-Boot image to the flash memory of the hardware. After building U-Boot, you have an ELF-format U-Boot binary executable file that contains debugger symbolic information. In addition, you have a U-Boot raw binary (.bin) file that you can write to flash memory on the target board.

2.1 Load XML File

To load a XML file with predefined flash programmer settings, follow these steps:

1. Select **Window > Show View > Other** from the CodeWarrior Workbench menu bar.
2. The **Show View** dialog box appears.
3. Expand the Debug tree control and select **Target Tasks**.
4. Click **OK**.

   The **Target Tasks** view appears.
5. Select the Root task group.
6. Right-click and select **Import** from the context menu that appears.

   The **Open** dialog box appears.
7. Select the appropriate xml file corresponding to the board from the following path:

   PA\bin\plugins\support\TargetTask\FlashProgrammer\QorIQ_P4\P4080DS_NOR_FLASH.xml

8. Click **Open** in the **Open** dialog box.
9. The target task with predefined flash settings is created and appears in the **Tasks** panel.
10. Double-click the task.

    The task opens in the **Flash Programmer Task** editor. This editor lets you configure the Flash Programmer target task.

2.2 Add a program action and a verify action

To add a Program action and a Verify action, follow these steps:

1. Select the predefined Flash Programmer target task.
2. In the Flash Programmer Task editor, click **Add Program/Verify Action**.

   The **Add Program/Verify Action** dialog box appears listing each flash device in the Flash Devices table.
3. Select the U-Boot raw binary (.bin) file that contains the data to be written to the flash device.
4. Specify the file path and name in the **File** text box or click the **Workspace**, **File System**, or **Variables** buttons to select the `u-boot.bin` file from the U-Boot folder you copied to your system.
5. From the **File Type** drop-down list, select the type of the source file. For `u-boot.bin`, select the **Binary** file type.
6. Check the **Erase sectors before program** checkbox. This option allows you to erase sectors before program so you don't need a separate action to erase the region where you will run the program.
7. Check the **Restrict to Addresses in this Range** checkbox.

   The **Start** and **End** text boxes activate.

   NOTE

   Write actions are permitted only within the specified address range.
8. In the **Start** text box, enter the start address of the flash memory range to program.
9. In the **End** text box, enter the end address of the flash memory range to program.
10. Check the **Apply Address Offset** checkbox.

    The **Address** text box activates. In the **Address** text box, enter the address offset as the value where you want U-Boot to be written to the flash device.
11. Click **Add Program Action**.

    The specified Program action is added to the Flash Programmer Actions table.
12. Click **Add Verify Action**.

    The specified Verify action is added to the Flash Programmer Actions table.
13. Click **Done**.

    The **Add Program/Verify Action** dialog box closes.

2.3 Run the Flash Programmer Target Task

To execute the Flash Programmer Target Task just configured, follow these steps:

1. Select the Flash Programmer Target Task to run from the **Target Tasks** view.

NOTE

To access the **Target Tasks** view select **Window > Show View > Debug > Target Tasks** from the main menu bar.

2. Start a debug session. For information on how to create a project and start a debug session, refer to the *Quick Start for PA 10 Processors* in *<CW_Installdir>\PA* folder.

NOTE

The run configuration of the Flash Programmer task is set to Active Debug Context by default so you need a active debug session to execute the task.

To run the target task with a launch configuration without an active debug session, you can change the run configuration to something different than "Active Debug Session". To change the default run configuration, right-click the task and select **Change Run Configuration** from the context menu.

3. The **Change Run Configuration** dialog box appears. From the **Run Configuration** drop-down list, select an appropriate run configuration.
4. Click the Execute icon in the **Target Task** view to run the selected Flash Programmer Target Task.

NOTE

You can monitor the results of the flash programmer actions in the **Console** view. Green indicates success, and red indicates failure. If RCW of the target board is erased, you can use JTAG configuration files to connect debugger to the target board. For details on RCW, see the reference manual for the target processor. For details on JTAG configuration files, refer the JTAG Configuration Files chapter in the PA Processors Targeting Manual. (Select **Help > Help Contents** to access the manual online).

3 Creating CodeWarrior Projects to Debug U-Boot and Kernel

To create a CodeWarrior Project to Debug U-Boot:

1. Start the CodeWarrior IDE.
2. Select **File > Import**. The **Import** wizard appears.
3. From the **CodeWarrior** container, select **Power Architecture ELF Executable**.
4. Click **Next**.
5. Specify the project name in the **Project name** text box.
6. Click **Next**.

7. Click **Browse** next to the **Executable** text box.
8. Select the U-Boot image or select the vmlinux file.

NOTE

In the **Select File** dialog box, from the **Files of type** list, select * to see the U-Boot and vmlinux executable files.

9. Click **Open**.
10. From the **Toolchain** list, select **Bareboard Application**.
11. Click **Next**.
12. From the **Processor** list, expand the processor family and select the required processor.
13. Click **Next**.
14. From the **Debugger Connection Types** list, select the required connection type.
15. From the Core index for this project list, select the required cores.
16. Click **Next**.
17. Specify the hardware settings, such as target hardware, connection type, and TAP address if you are using Ethernet or Gigabit TAP.
18. Click **Finish**.

    The imported project appears in the **CodeWarrior Projects** view.

    You just finished creating a CodeWarrior project to debug the U-Boot or kernel image.

4 Debugging the U-Boot or Kernel Projects

For information on how to configure and debug the CodeWarrior project you just created for U-Boot or Kernel images, please refer to the PA Processors Targeting Manual. (Select **Help** > **Help Contents** to access the manual online).

Appendix A Images folder

All images built by System Builder are put in *sb_work/build_<bsp><bit-type>_release/deploy/glibc/images/*. Table 1-1 lists the typical directory/image files:

NOTE

The exact contents of the image files depend on the setting of the `IMAGE_FSTYPES` variable.

Table 4-1. System Builder Image Files

Image Files	Usage
sb_work/build_<bsp>_release/deploy/glibc/images/	image directory
sb_work/build_<bsp>_release/deploy/glibc/bin/	host scripts/tools directory

Table 4-1. System Builder Image Files

uImage-<bsp>.bin	kernel image that can be loaded with U-Boot
devel-image-<bsp>.ext2.gz.u-boot	maximum size ramdisk image that can be loaded with U-Boot
devel-image-<bsp>.ext2.gz	gzipped ramdisk image
devel-image-<bsp>.tar.gz	gzipped tar archive of the image
u-boot-<bsp>.bin	U-Boot binary image that can be programmed into board Flash
<bsp>.dtb	device tree binary (dtb)
fsl_fman_ucode_P4080_92_8.bin	fman ucode for P4080 rev1 board
fsl_fman_ucode_P3_P4_P5_101_8.bin	fman ucode for P3041/P4080 rev2 board/P5020
boot/hv.uImage	uImage for hypervisor
hv-cfg/*/*/hv.dtb	dtb for hypervisor
rcw/*/rcw_*.bin	rcw for hypervisor

Appendix B Values of <bsp> for supported targets

Table 4-2. <bsp> values for supported targets

Targets	**<bsp> Values**
P1023RDS	p1023rds
P3041DS	p3041ds
P4080DS	p4080ds
P5020DS-32b	p5020ds-32b

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4436

April 5, 2012