Freescale Semiconductor
Application Note

Document Number: AN5137

Debugging ARMv7 Applications in Environment Initialized by U-Boot / ROM Target Debug

1. Introduction

This document describes the necessary steps required to use CodeWarrior for QorIQ LS series - ARM V7 ISA for debugging applications running in an environment initialized by U-Boot.

This document explains:

- How to build U-Boot to add bootelf support
- How to load the application to the target
- How to debug from the entry point using CodeWarrior, an application started using the `bootelf` command
- How to debug from the entry point using CodeWarrior, an application flashed in NOR
- How to debug a secure ROM target application
- How to debug ROM applications on some specific LS1 boards

Contents

2. Debugging applications started from U-Boot

2.1. Add bootelf support to U-Boot

The *bootelf* command allows booting an ELF image in memory after the U-Boot is loaded. To add bootelf support to U-Boot, perform these steps:

1. Install SDK with Yocto and build U-Boot. For more details, see Freescale Infocenter.

   ```
   $ bitbake u-boot
   ```

2. Go to the U-Boot source location and edit *config_cmd_default.h*, to add the *bootelf* command.

Figure 1. Edit config_cmd_default.h file

```
/*
 * Alphabetical list of all commands that are configured by default.
 * This is essentially all commands minus those that are considered
 * "non-standard" for some reason (memory hogs, requires special
 * hardware, not fully tested, etc.).
 */

#define CONFIG_CMD_BDI      /* bdinfo           */
#define CONFIG_CMD_BOOTD    /* bootd            */
#define CONFIG_CMD_ELF      /* ELF (VxWorks) load/boot cmd  */
#define CONFIG_CMD_CONSOLE  /* coninfo          */
#define CONFIG_CMD_ECHO     /* echo arguments       */
```

3. Rebuild U-Boot:

   ```
   $ bitbake -c compile -f u-boot
   ```

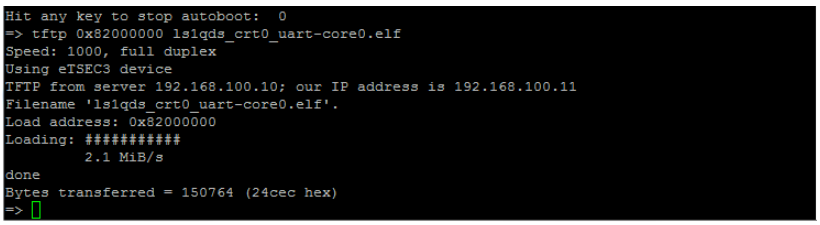
   This adds bootelf support to the U-Boot image available in the u-boot folder.

2.2. Run and debug application

First, you need to create an ARMv7 application, starting from stationary project. The project must be created with UART I/O support, to display messages on U-Boot console. The steps are:

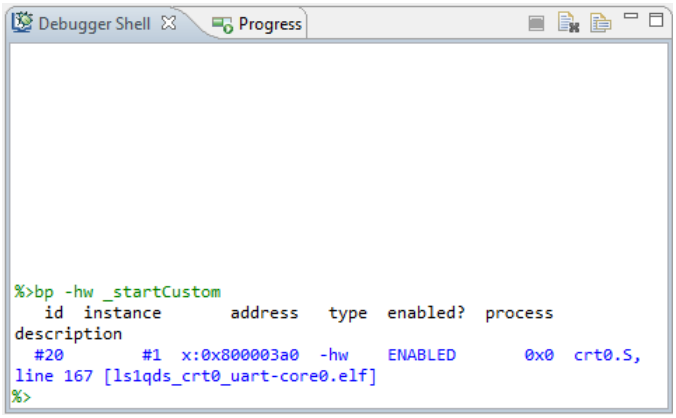
1. Create an ARMv7 project and build it using a RAM target.
2. Power on the board and stop at U-Boot prompt.
3. Copy the ELF file to a TFTP server location.
4. Load the ELF to RAM, using U-Boot commands.

Figure 2. Load ELF to RAM



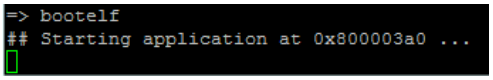
5. To start debugging from the entry point, connect to the target board using Attach launch configuration and set a hardware breakpoint at *_startCustom*, using Debugger Shell command *bp -hw _startCustom*.

Figure 3. Set a hardware breakpoint at _startCustom



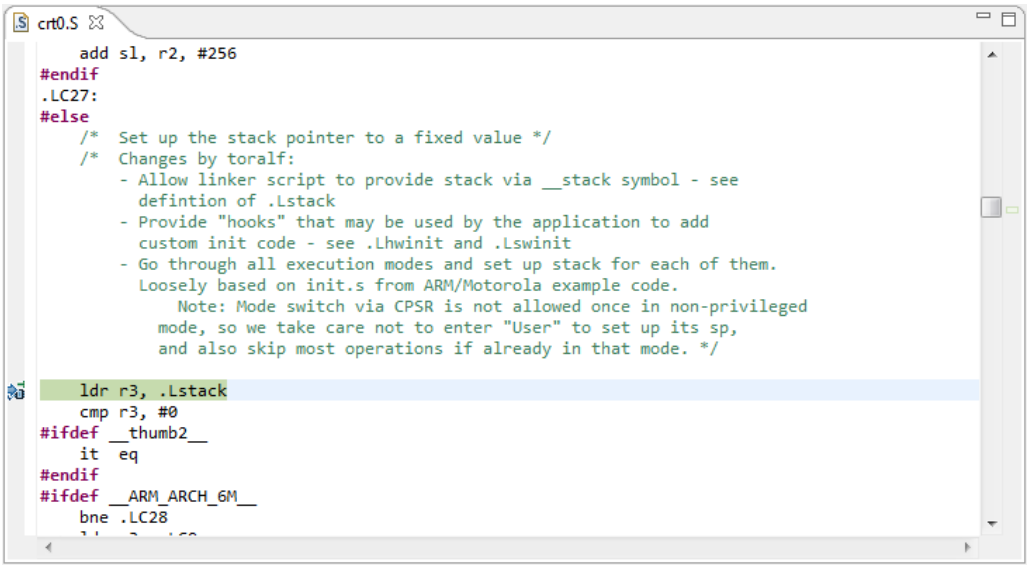
6. From U-Boot prompt, run the *bootelf* command. This executes the ELF from RAM.

Figure 4. Run bootelf command



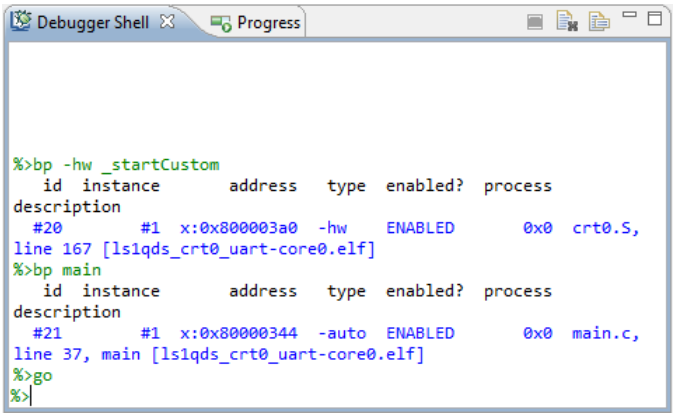
7. Breakpoint will be hit and you can perform debugging from the entry point.

Figure 5. Perform debugging from entry point



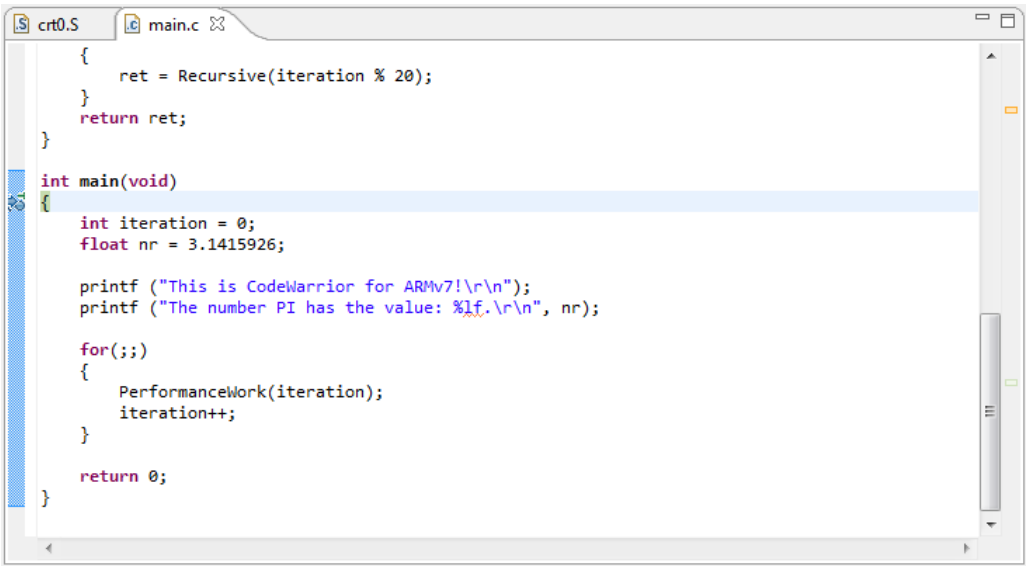
8. To continue debugging from *main* function, set a breakpoint using Debugger Shell command *bp main* and resume the core.

Figure 6. Set a breakpoint at main function



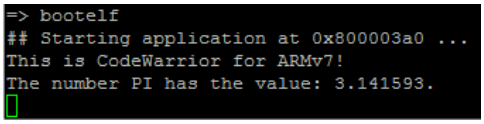
Breakpoint will be hit, as shown in the figure below.

Figure 7. Application stopped at main function



9. Continue debugging (step, run, or breakpoint) till the end of the application.

Figure 8. Debug application



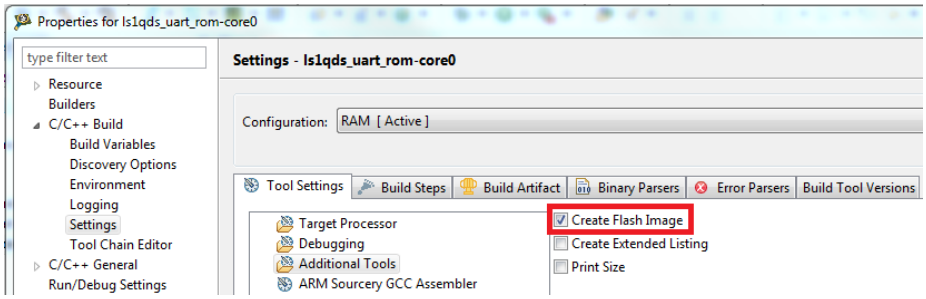
NOTE	To debug the application from the entry point, a copy of the startup file (*crt0.S*) needs to be included in source form. This is not applicable to CodeWarrior for ARM v7 10.0.3 release.

3. Debugging applications flashed in NOR

To debug an application flashed in NOR, perform these steps:

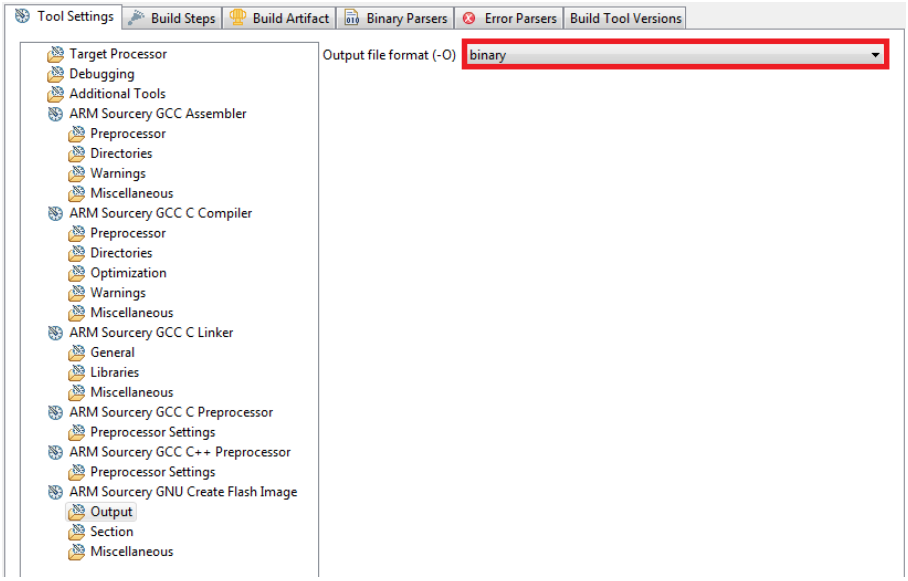
1. Create an ARMv7 project.
2. Choose **Project > Properties** to open the **Properties** window for the project.
3. Select **C/C++ Build > Settings** to open project settings.
4. On the **Tool Settings** tab, select **Additional Tools** to open the **Additional Tools** page.
5. Select the **Create Flash Image** checkbox, as shown in the figure below.

Figure 9. Select Create Flash Image option

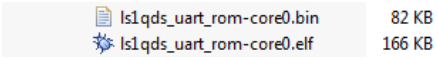


6. On the **Tool Settings** tab, select **ARM Sourcery GNU Create Flash Image > Output** to open the **Output** page.
7. Choose **binary** as the output file format, as shown in the figure below.

Figure 10. Choose output file format

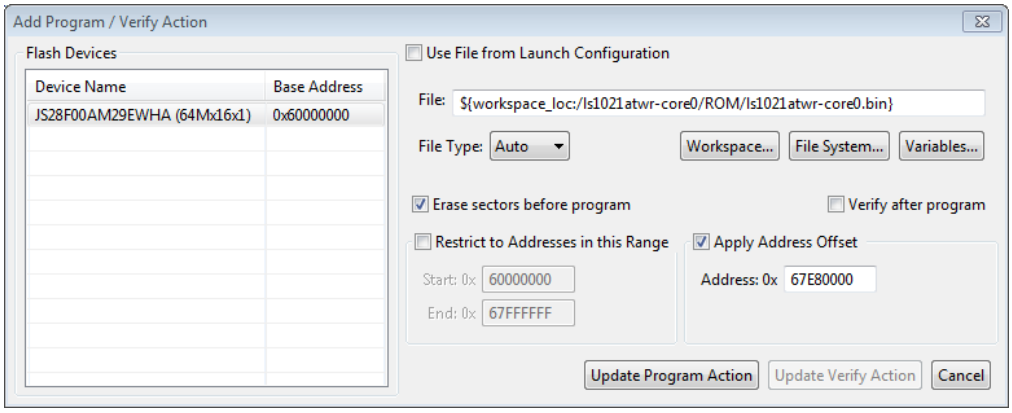


8. Click **Apply**, and then click **OK**.
9. Build the application using ROM target. Beside the ELF file, a binary file will be produced.



10. Connect to the target board and load the application binary file to NOR flash, using Flash Programmer, as shown in the figure below.

Figure 11. Add Program / Verify Action dialog



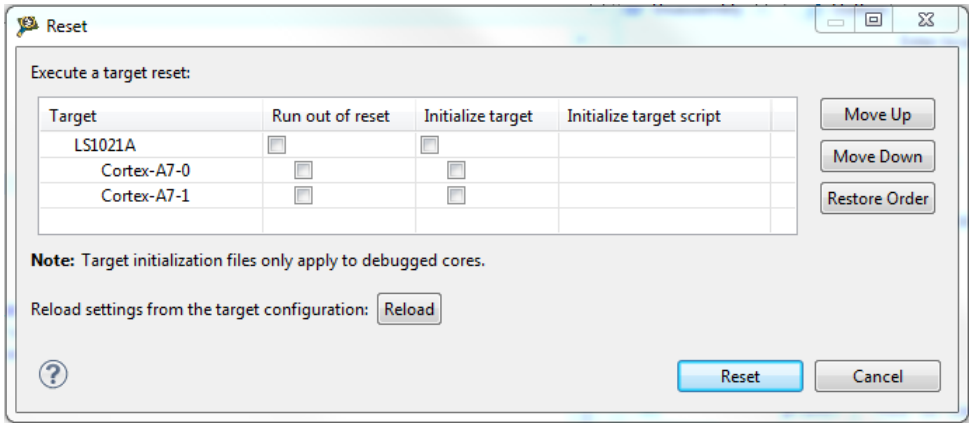
NOTE The binary file will be flashed in NOR at the same address where U-Boot is usually placed. Because of this, if U-Boot is present on the target board, it will be overwritten by the application. The address is specified in the ROM linker file.

At the board reset, the application will run and it can be debugged.

```
This is CodeWarrior for ARMv7!
This is the ROM target!
The number PI has the value: 3.141593.
```

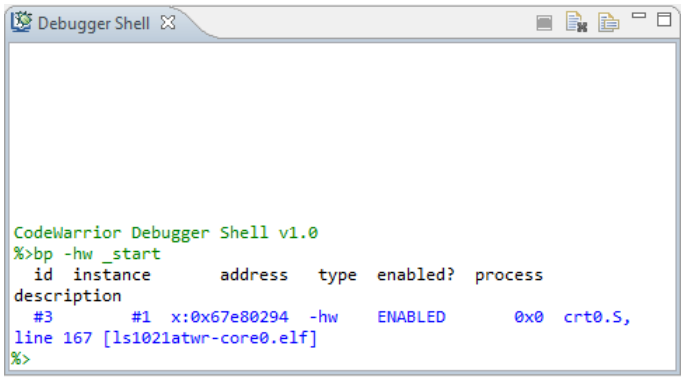
11. To start debugging from the entry point, connect to the target board using ROM Attach launch configuration, and then reset the board with no initialization file.

Figure 12. Reset dialog



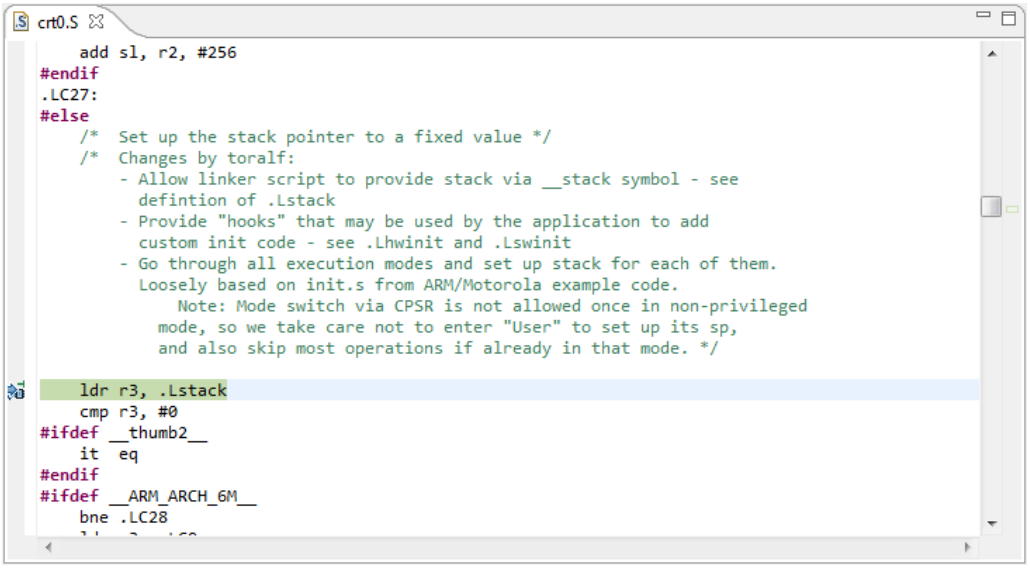
12. Set a hardware breakpoint at `_start`, using Debugger Shell command `bp -hw _start`, as shown in the figure below.

Figure 13. Set a hardware breakpoint at _start



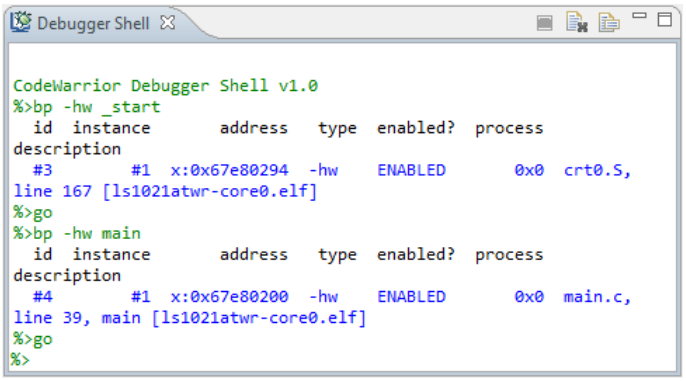
13. Resume the core. Breakpoint will be hit and you can perform debugging from the entry point.

Figure 14. Perform debugging from entry point



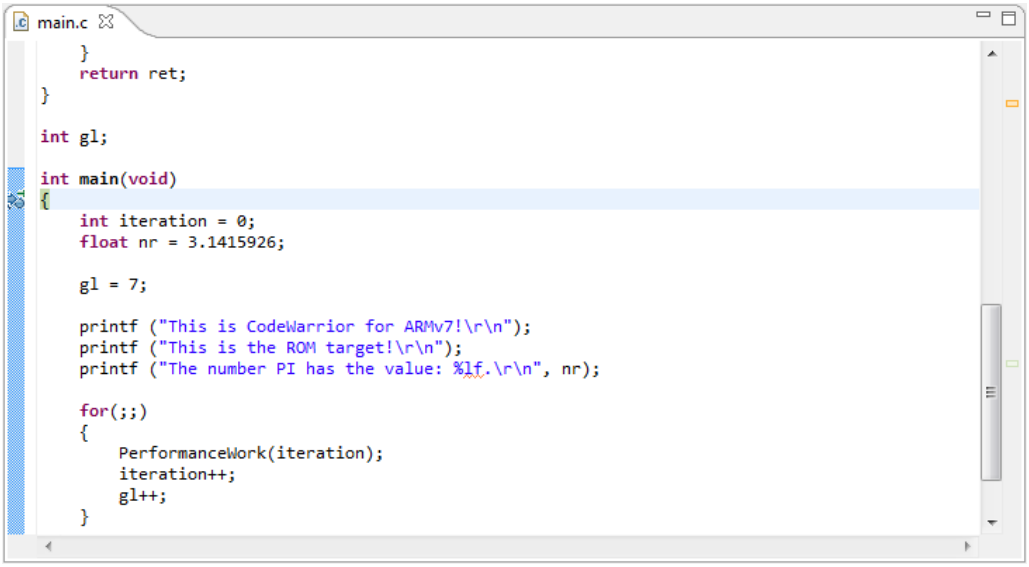
14. To continue debugging from the *main* function, set a breakpoint using Debugger Shell command *bp main* and resume the core.

Figure 15. Set a breakpoint at main function

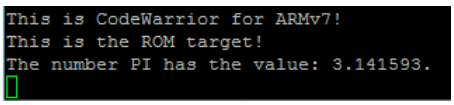


Breakpoint will be hit, as shown in the figure below.

Figure 16. Application stopped at main function



10. Continue debugging (step, run, or breakpoint) till the end of the application.



NOTE	To debug the application from the entry point, a copy of startup file (`crt0.S`) needs to be included in source form. In addition, the LCF file needs to be modified to include changes necessary for application to be loaded to ROM. This is not applicable to CodeWarrior for ARM v7 10.0.3 release.

4. Debugging secure ROM target applications

Debugging a ROM target application signed with Code Signing Tool (CST) involves, besides the steps from Debugging applications flashed in NOR, signing the application and generating the application header. For details on CST, see Freescale Infocenter.

To sign a ROM target application with CST, perform these steps:

1. Generate private key - public key pair.
2. Copy ROM target application to */tmp/sysroots/x86_64-linux/usr/bin/cst*.
3. Specify the name of the application and command sequence file (CSF) header in the *input_files/uni_sign/ls1/input_uboot_nor_secure* file.

Figure 17. Specify application name and CSF header

```
------------------------------------------------------
# Specify IMAGE, Max 8 images are possible. DST_ADDR is required only for Non-PBL Platform. [Mandatory]
# USAGE : IMAGE_NO = {IMAGE_NAME, SRC_ADDR, DST_ADDR}
IMAGE_1={ls1021_rom.bin,67f80000,ffffffff}
IMAGE_2={,,}
IMAGE_3={,,}
IMAGE_4={,,}
IMAGE_5={,,}
IMAGE_6={,,}
IMAGE_7={,,}
IMAGE_8={,,}
------------------------------------------------------
# Specify OEM AND FSL ID to be populated in header. [Optional]
# e.g FSL_UID=11111111
FSL_UID=
FSL_UID_1=
OEM_UID=
OEM_UID_1=
------------------------------------------------------
# Specify the file names of csf header and sg table. (Default :hdr.out) [Optional]
OUTPUT_HDR_FILENAME=hdr_ls1021_rom.out
```

4. Execute the *./uni_sign input_files/uni_sign/ls1/input_uboot_nor_secure* command.
5. Connect to the target board and load the application binary file and CSF header to NOR flash, using Flash Programmer.
6. Start debugging (step, run, or breakpoint) using the steps from Debugging applications flashed in NOR.

NOTE	For secure boot in RCW, set SB_EN = 1. Also, PBI commands must contain reference to CSF header address. The binary file will be flashed in NOR at the same address where U-Boot is usually placed. Because of this, if U-Boot is present on the target board, it will be overwritten by the application.

5. Debugging ROM target applications - Use cases

5.1. Debugging an LS102MARDB ROM target application

To debug a ROM application running on an LS102MARDB target, perform these steps:

1. Create an ARMv7 project for LS102MARDB with ROM Attach launch configuration.
2. Set the board for I2C boot (SW1[6:7]).
3. Power on the board and interrupt the autoboot sequence.
4. Connect to the target board using ROM Attach launch configuration. If the target is in the Running mode, then suspend the target.
5. Set the PC at the value 0x20000000 (this is the entry point of application) using Debugger Shell command *reg PC=0x20000000*.
6. Start debugging (step, run, or breakpoint).

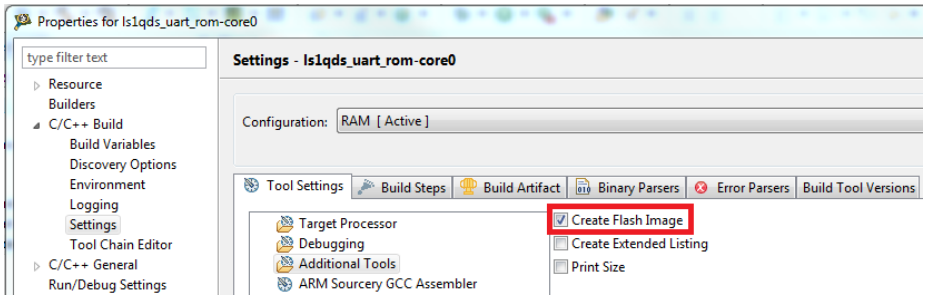
NOTE	After the application is built using CodeWarrior, use Flash Programmer to write the application to ROM, ensuring that the **Apply Address Offset** checkbox is not selected.

5.2. Debugging an LS1024ARDB ROM target application

To debug a ROM application running on an LS1024ARDB target, follow these steps:

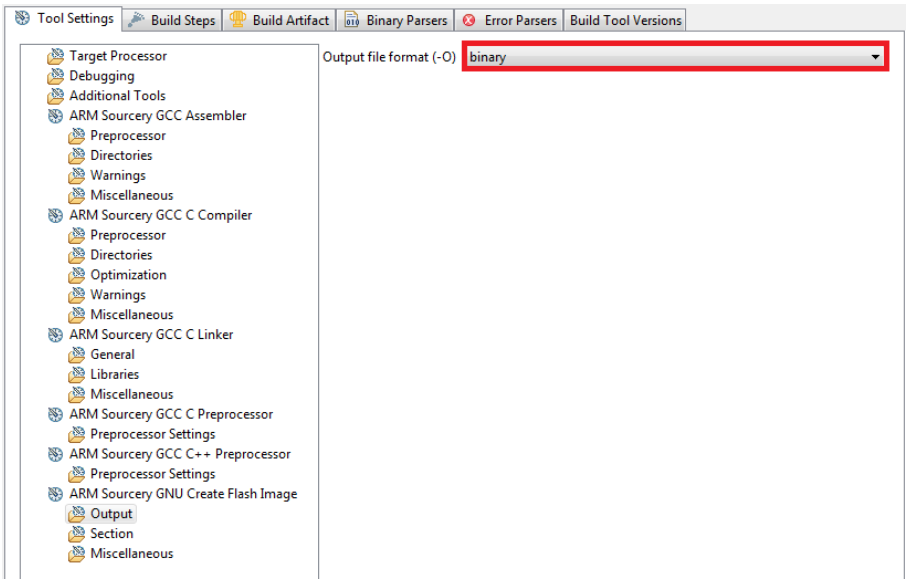
1. Create an ARMv7 project.
2. Ensure that the LS1024ARDB SDK is available.
3. Choose **Project > Properties** to open the **Properties** window for the project.
4. Select **C/C++ Build > Settings** to open project settings.
5. On the **Tool Settings** tab, select **Additional Tools** to open the **Additional Tools** page.
6. Select the **Create Flash Image** checkbox, as shown in the figure below.

Figure 18. Select Create Flash Image option



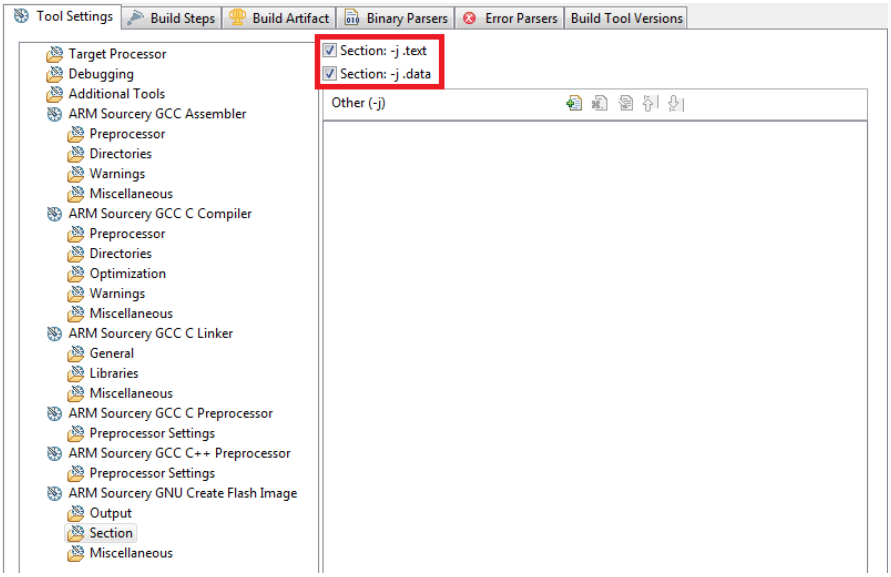
7. On the **Tool Settings** tab, select **ARM Sourcery GNU Create Flash Image > Output** to open the **Output** page.
8. Choose **binary** as the output file format, as shown in the figure below.

Figure 19. Choose output file format



9. On the **Tool Settings** tab, select **ARM Sourcery GNU Create Flash Image > Section** to open the **Section** page.
10. Select the checkboxes labeled **Section: -j .text** and **Section: -j .data**.

Figure 20. Select Section page options



11. Click **Apply**, and then click **OK**.
12. Build the application using ROM target. Besides the ELF file, a binary file will be produced.
13. Rename the binary image to *uImage* and place it in the */target/linux/comcerto2000/image/ImageGenarator/* folder of the LS1024ARDB SDK.

14. Sign the application using *kernel_gen.sh* script. Singing the application will make it recognized by the microloader.
    A new *uImage1* will be produced.
15. Connect to the target board and using Flash Programmer, load the application binary file to NOR flash, at address *0xC0020000*.
16. Reset the board and microloader will recognize and run the application.
17. Start debugging (step, run, or breakpoint).

NOTE To debug the application from the entry point, connect to the target after reset, suspend the target, and set the PC at the value *0x01000000*, using Debugger Shell command *reg PC=0x01000000*.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

Document Number: AN5137

9 July 2015