Freescale Semiconductor
Application Note

Document Number: AN4956

U-Boot/Barebox Debug using CodeWarrior for QorIQ LS series - ARM V7 ISA

1. Introduction

This document describes the steps required for U-Boot/Barebox debugging using CodeWarrior Development Studio for QorIQ LS series - ARM V7 ISA.

This document includes the following sections:

- Preliminary background
- Creating an ARMv7 project
- Debugging U-Boot
- Debugging U-Boot SPL
- Debugging Barebox
- Downloading U-Boot/Barebox binary on target board
- Calculating PIC load address for U-Boot DDR relocation

Contents

2. Preliminary background

This section describes the steps required to compile U-Boot/Barebox for the LS1 boards.

2.1. Download SDK

To debug U-Boot/Barebox using CodeWarrior, download the latest SDK for QorIQ from www.freescale.com.

2.2. Compile U-Boot/Barebox

U-Boot/Barebox binary must be built with debug information to be debugged using CodeWarrior.

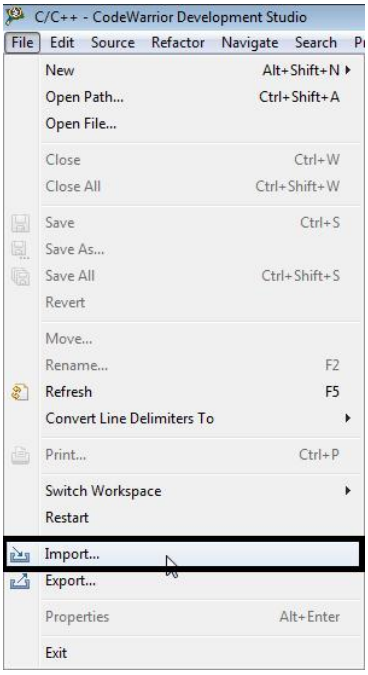
Also, U-Boot/Barebox binary must be downloaded on the target board and must be from the same build as U-Boot/Barebox image. See Downloading U-Boot/Barebox binary on target board for details on how to download U-Boot/Barebox binary on the target board.

3. Creating an ARMv7 project

To create an ARMv7 bare-metal project for U-Boot/Barebox debug, follow these steps:

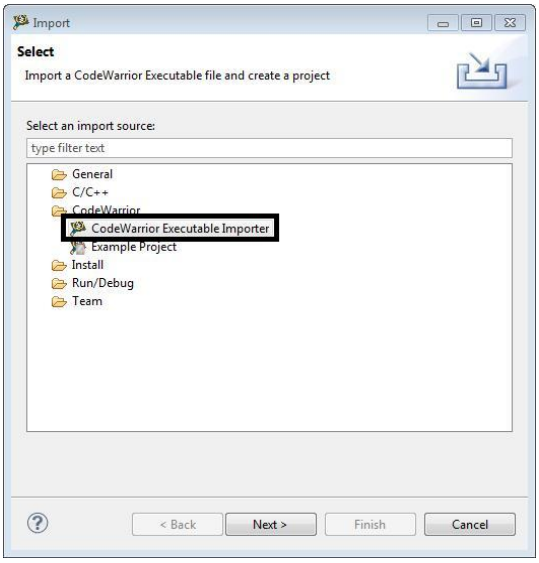
1. Start CodeWarrior for QorIQ LS series - ARM V7 ISA.
2. Choose **File** > **Import** to import the U-Boot/Barebox executable file generated during the U-Boot/Barebox compilation. It can be found in the `U-Boot/Barebox` folder.

Figure 1. CodeWarrior File menu



3. Choose the source to import and click **Next**.

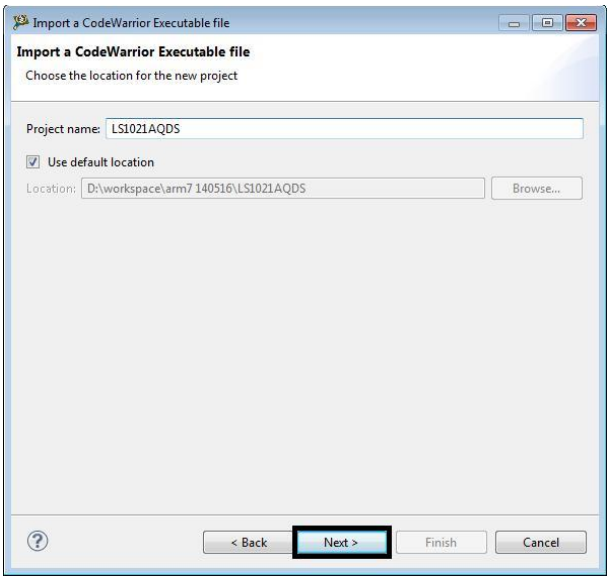
Figure 2. Import dialog



The **Import a CodeWarrior Executable file** wizard starts, as shown in the figure below.

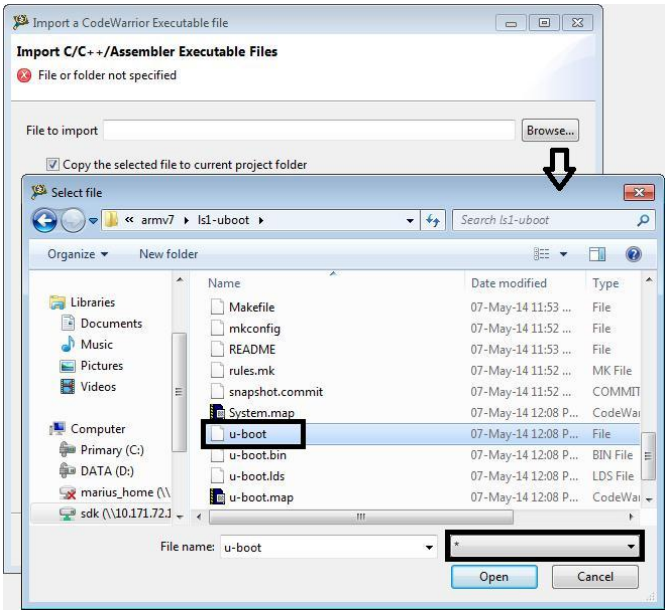
4. Specify project name and location, or use the default location and click **Next**.

Figure 3. Import a CodeWarrior Executable file page



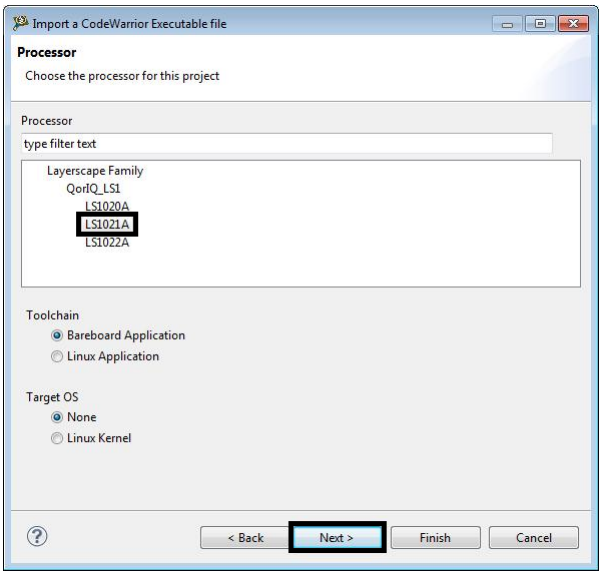
5. Browse to the U-Boot/Barebox executable file and click O**pen**. By default, CodeWarrior looks for an `.elf` extension; therefore, change the file type in the lower-right corner of the **Select file** dialog.

Figure 4. Select U-Boot executable file



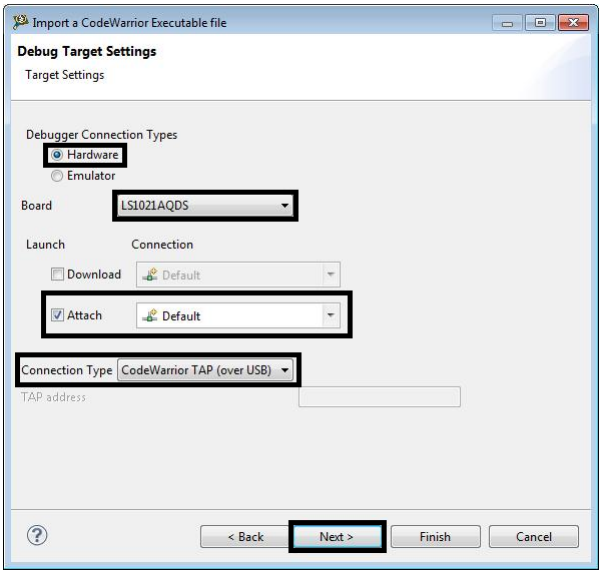
6. Select processor type for the project and click **Next**.

Figure 5. Processor page



7. Select the debugger connection type, board, launch configuration, and connection type, and click **Next**.

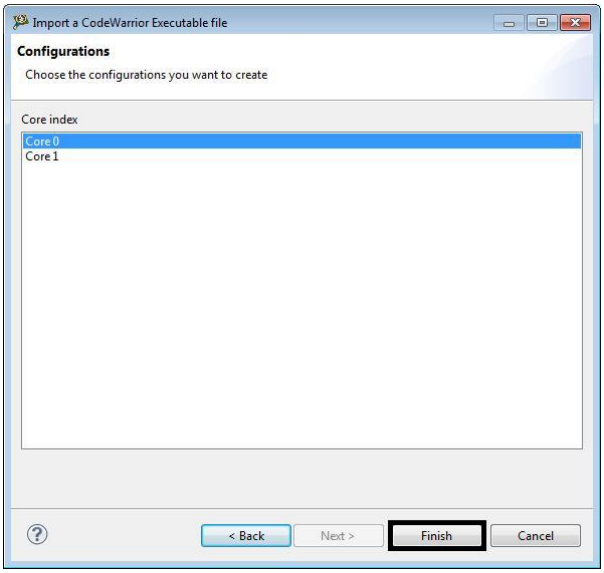
Figure 6. Debug Target Settings page



NOTE By default, U-Boot is generated as a *shared object file*, and not as an *executable file*. In this case, the Download launch configuration does not work; therefore, you need to use the Attach launch configuration.
If U-Boot is not available on the target board, then Flash Programmer should be used to program U-Boot on the target board.

8. Choose the configurations you want to create, and then click **Finish** to close the wizard.

Figure 7. Configurations page



4. Debugging U-Boot

When U-Boot starts, it is running from ROM space. However, running from flash would make it almost impossible to read from flash while executing code from flash, not to mention updating the U-Boot image in flash itself. To be able to do that, U-Boot relocates itself into RAM. Because of this, we have two phases with different program addresses. The following sections show how to debug U-Boot in both phases.

4.1. U-Boot debug before relocation

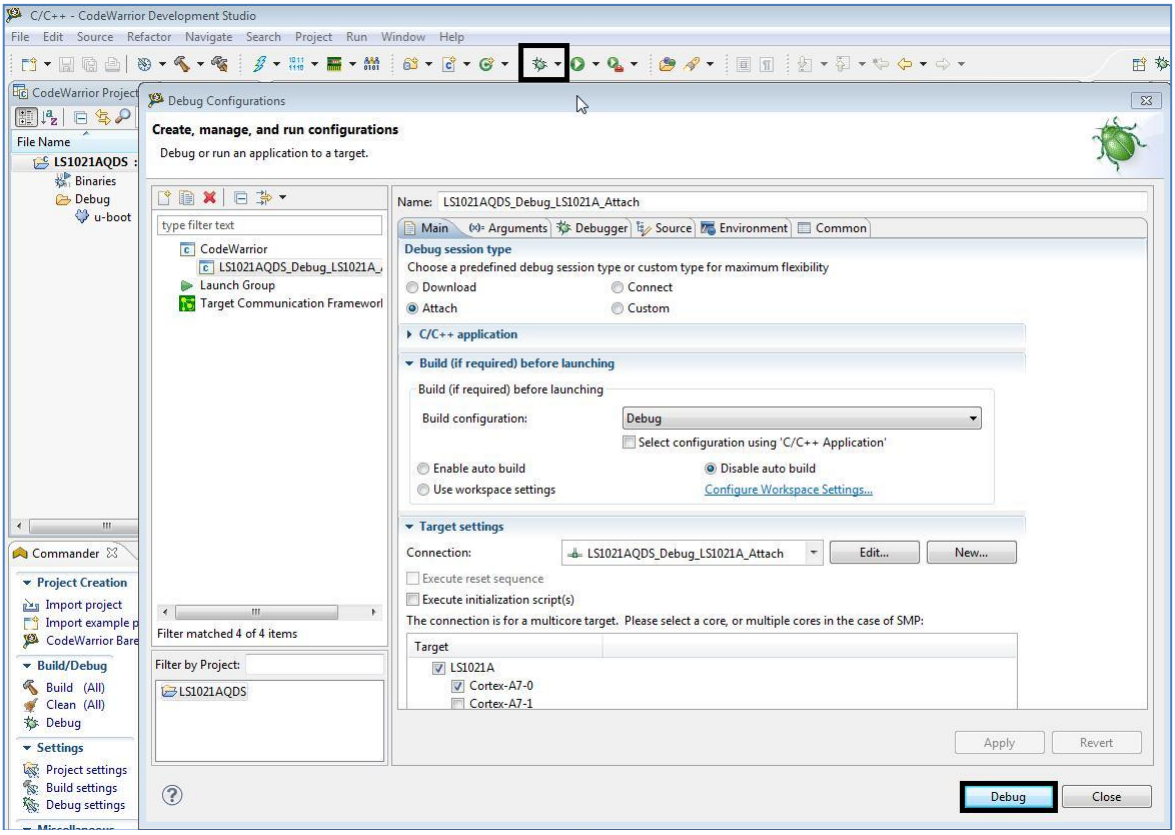
Before U-Boot relocation, the addresses from the ELF file can be used as it is.

The U-Boot executable file generated during the U-Boot compilation should be imported as a CodeWarrior project (for more information, see Creating an ARMv7 project).

After the CodeWarrior project is created, perform these steps to start U-Boot debug:

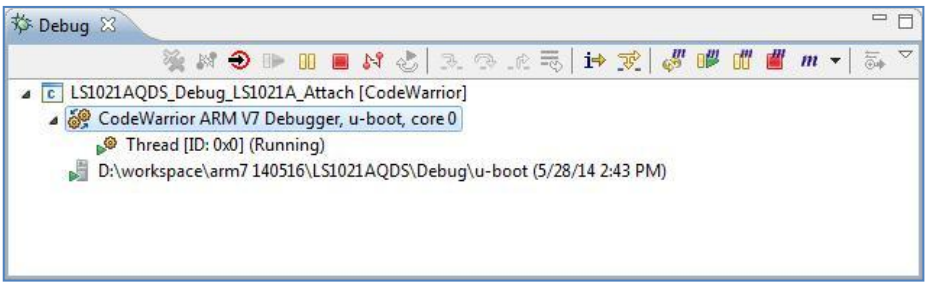
1. Choose **Run** > **Debug configurations** to open the **Debug Configurations** dialog, and click **Debug** (see figure below).

Figure 8. Debug Configurations dialog



The connection initializes and configures the TAP, and then it will attach to the board (see figure below).

Figure 9. Debug view

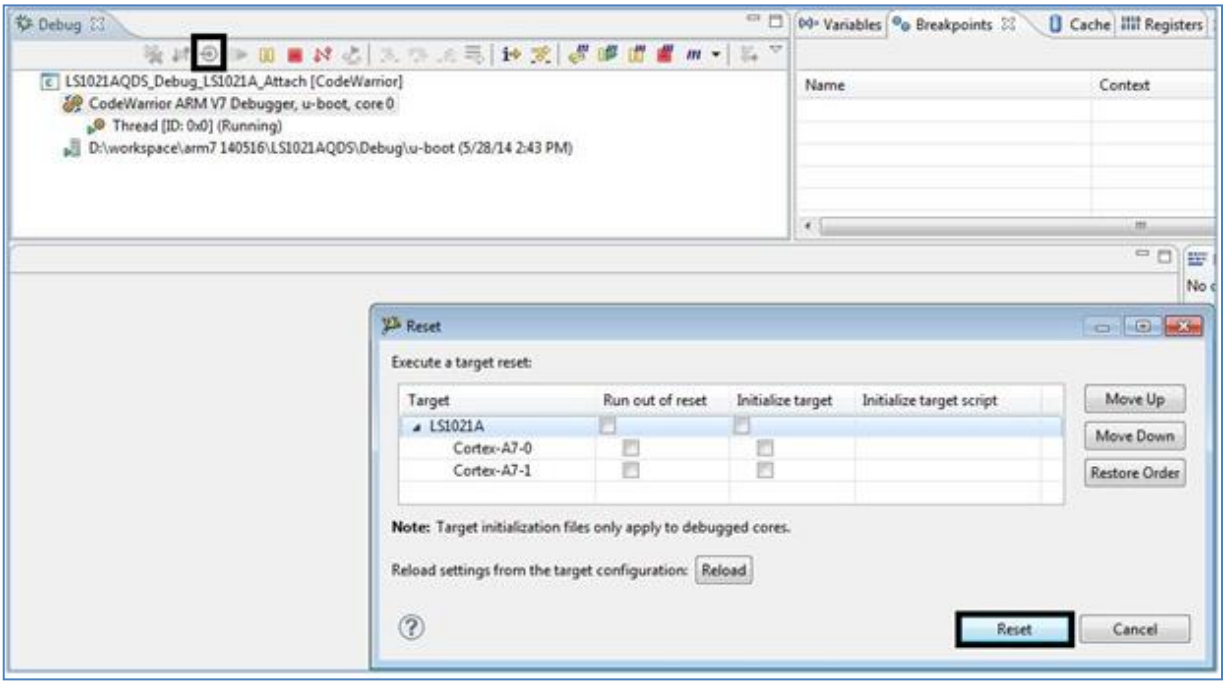


2. To reinitialize the target from CodeWarrior, click the **Reset** icon in the **Debug** view toolbar.

   The **Reset** dialog opens (see figure below).

3. Ensure that no initialization file is selected in the **Reset** dialog and click **Reset**.

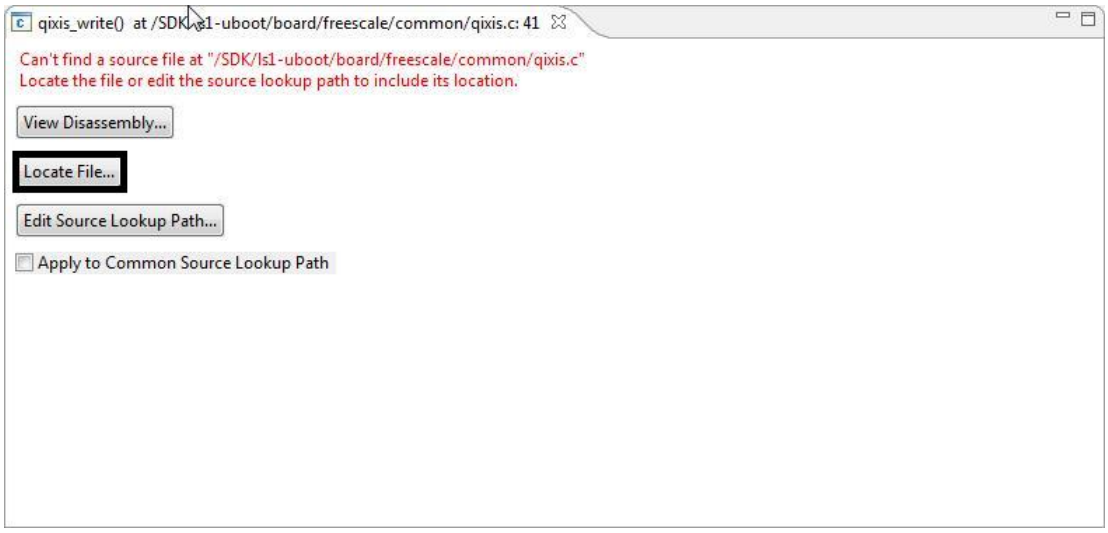
Figure 10. Reset dialog



After reset, debugger will prompt for source location, as shown in the figure below.

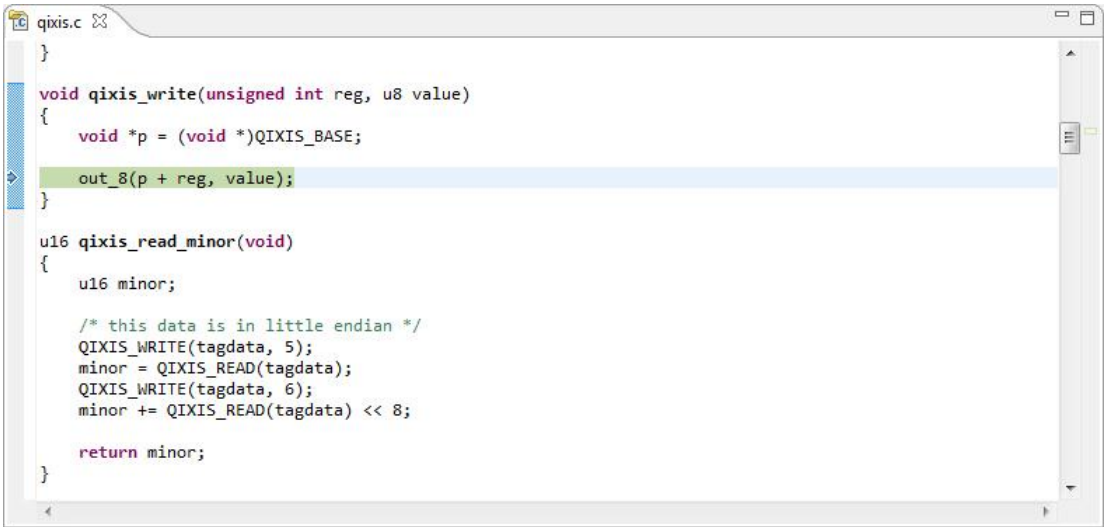
4. Click **Locate File** and specify file path.

Figure 11. Specify file location



After the path is provided, source will become available in CodeWarrior.

Figure 12. File editor



5. Set a hardware breakpoint at *_start*, using Debugger Shell command *bp –hw _start*.

Figure 13. Set a hardware breakpoint at _start

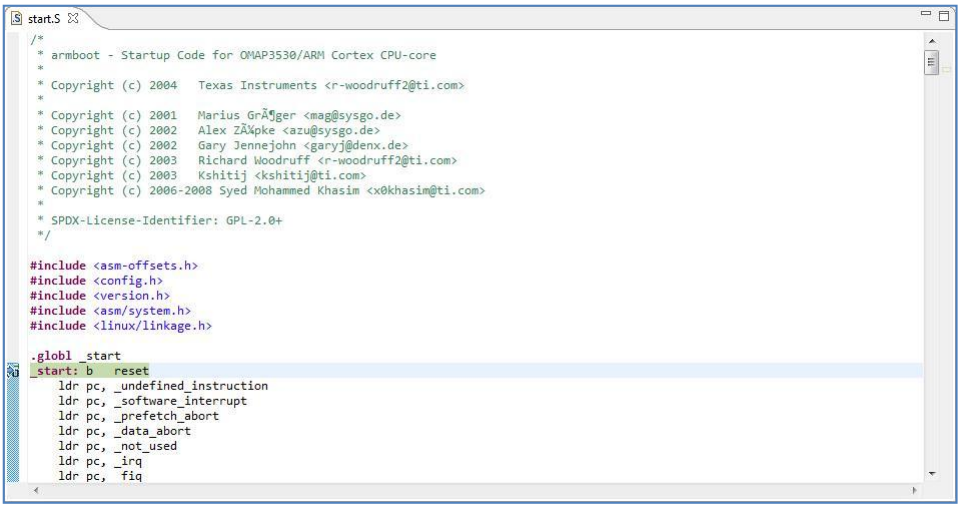
NOTE Hardware breakpoint must be used before DDR initialization.

6. Resume debugging using **F8** or Debugger Shell command *go*.

Figure 14. Perform debugging

7. Breakpoint will be hit and U-Boot debugging can be performed from *_start*.

Figure 15. File editor



The next section describes how to perform U-Boot debug after relocation in RAM.

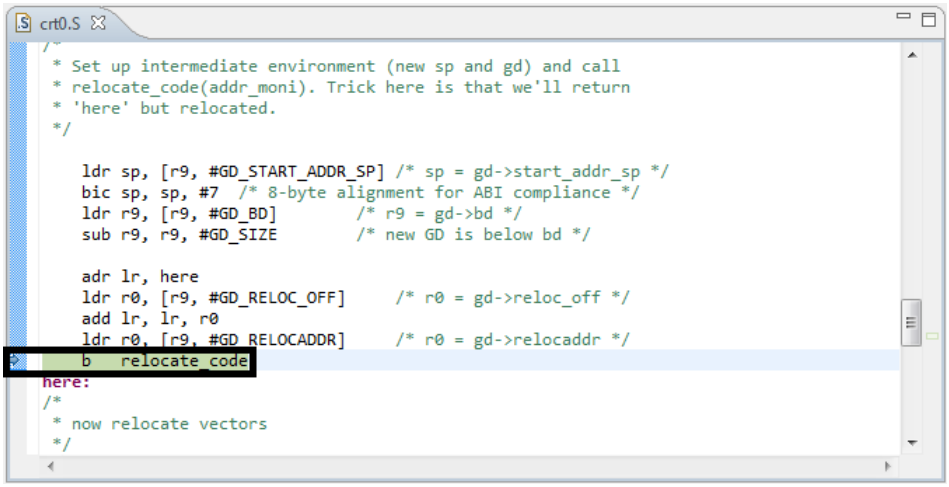
NOTE	If you encounter reset skid issue, the program will not stop at *_start* symbol. As a workaround, you can set a hardware breakpoint at *_start*, and move PC to *_start* symbol address. This issue has been resolved in FPGA v11 image for the LS1 QDS board, but it is present for the LS1 TWR board using CMSIS-DAP probe.

4.2. U-Boot debug after relocation

For U-Boot debugging after relocation, you need to know the address U-Boot relocates itself to. Perform these steps to find out the relocation address:

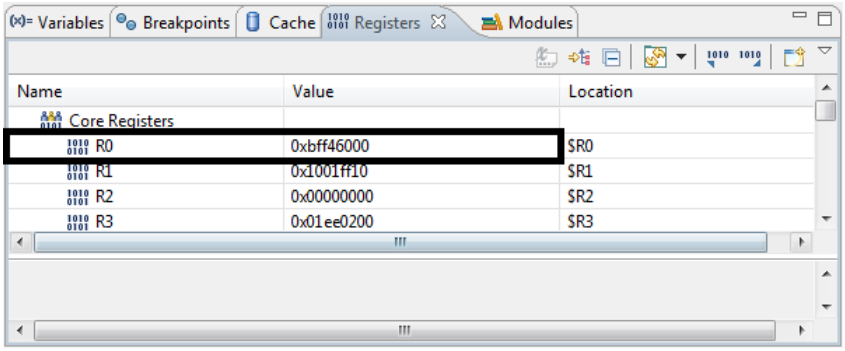
1. Set a hardware breakpoint at *_main*. Resume debugging using **F8** or Debugger Shell command *go*.
2. Debug until *b relocate_code* (see figure below).

Figure 16. Perform debugging



3. The relocation address is stored in R0 register. Open **Registers** view and read the value for R0 register (see figure below).

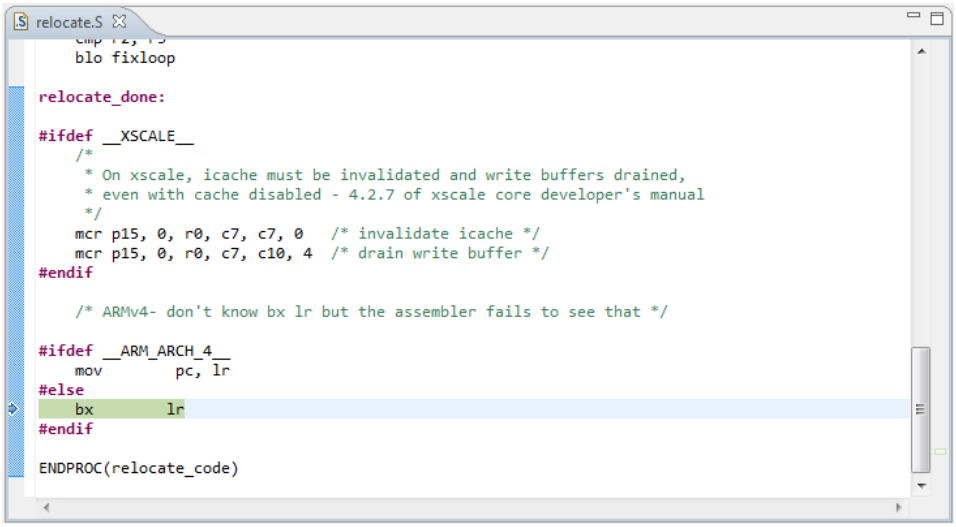
Figure 17. Registers view



For U-Boot debug after relocation, perform these steps:

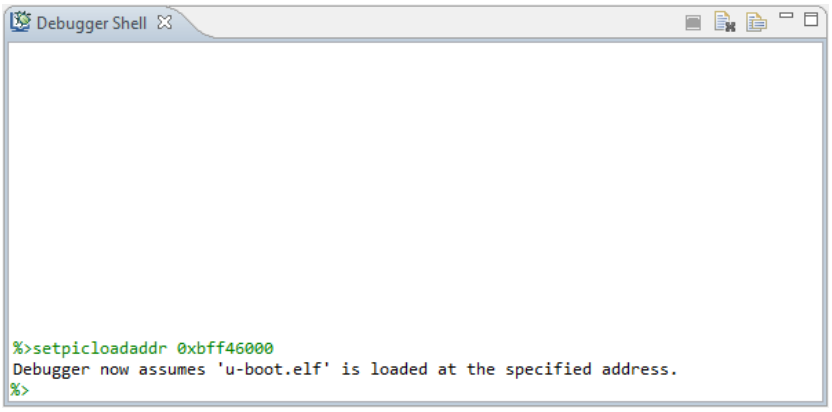
1. Open the *relocate.S* file. In this file, the last instruction before completion of U-Boot relocation is *bx lr*.

Figure 18. File editor



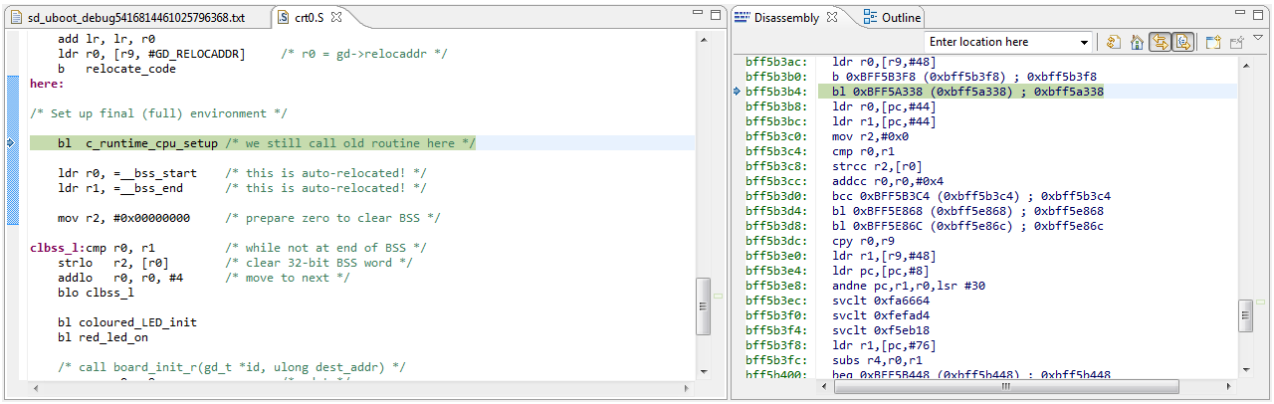
2. Step into *bx lr*. Only disassembly will be available.
3. Instruct the debugger to reload the symbols with position independent code (PIC) load address.
4. Set the PIC load address to *0xBFF46000* using the Debugger Shell command *setpicloadaddr 0xBFF46000*.

Figure 19. Debugger Shell view



The symbols are loaded and debugging (step, run, or breakpoint) can be done until the U-Boot boots up (see figure below).

Figure 20. File editor and Disassembly view



NOTE The relocation address can be read from U-Boot prompt using *bdinfo* command. In case the relocation address is not correct when reading R0, then to calculate the PIC load address after U-Boot relocation, see Calculating PIC load address for U-Boot DDR relocation.

5. Debugging U-Boot SPL

For the situations when U-Boot is located in the NAND/SPI/SD card (flash devices that are not memory mapped), first load U-Boot SPL to initialize the hardware, and then load the U-Boot image. The U-Boot SPL executable file generated during U-Boot compilation should be imported as a CodeWarrior project (for more information, see Creating an ARMv7 project).

After creating the CodeWarrior project, debug U-Boot SPL using the steps provided in U-Boot debug before relocation.

6. Debugging Barebox

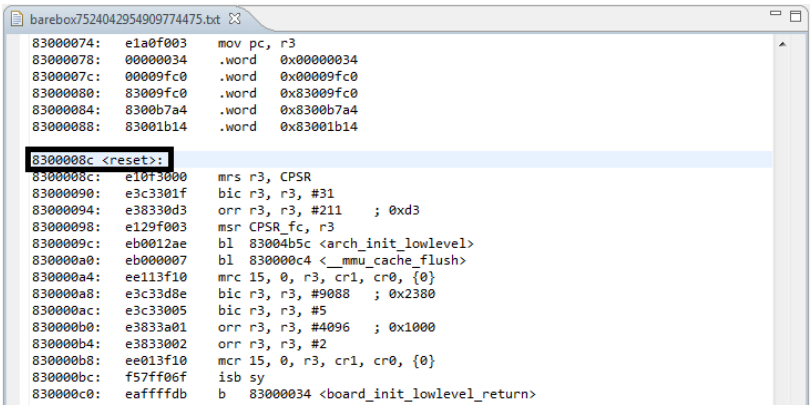
Barebox is an alternative bootloader supported by LS1024A target. The Barebox executable file generated during Barebox compilation should be imported as a CodeWarrior project (for more information, see Creating an ARMv7 project).

After the CodeWarrior project is created, perform these steps to start Barebox debug:

1. Choose **Run** > **Debug configurations** to open the **Debug Configurations** dialog, and click **Debug**.
2. To debug Barebox from reset address, reinitialize the target from CodeWarrior by clicking the **Reset** icon in the **Debug** view toolbar.

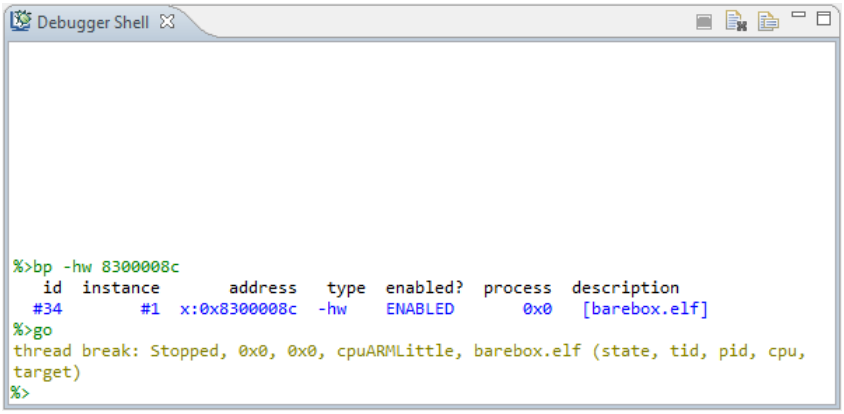
3. Find the reset address of the microloader. To do this, first import the microloader image in CodeWarrior and disassemble it, and then search for *<reset>* in the disassemble file (see figure below).

Figure 21. Find reset address of microloader



4. Set a hardware breakpoint at the reset address of the microloader, as shown in the figure below.

Figure 22. Set a hardware breakpoint at reset address



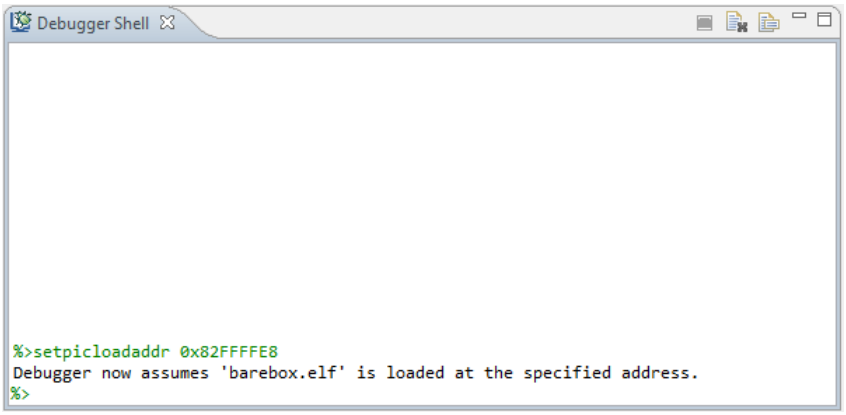
5. Resume debugging using **F8** or Debugger Shell command *go*.

   Breakpoint will be hit and only disassembly will be available.

NOTE	To have the mapping between the sources and the code, debugger must be instructed to reload the symbols with position independent code (PIC) load address. To calculate the PIC load address, disassemble Barebox executable to obtain the reset address. Calculate the difference between *reset address from microloader executable* and *reset address from Barebox executable*.

5. Set the PIC load address to *0x82FFFFE8* using the Debugger Shell command *setpicloadaddr 0x82FFFFE8*, as shown in the figure below.

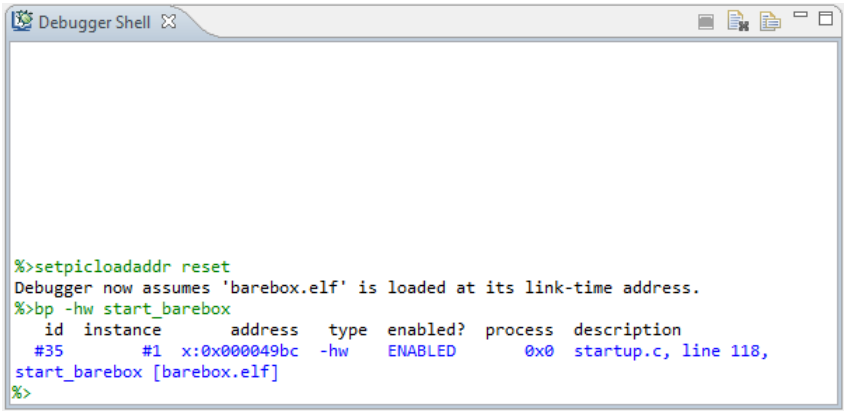
Figure 23. Set PIC load address



The symbols are loaded and debugging (step, run, or breakpoint) can be done until the Barebox DDR relocation.

6. To continue debugging after DDR relocation, reset the PIC load address using the Debugger Shell command *`setpicloadaddress reset`*, and set a hardware breakpoint at *`start_barebox`*, as shown in the figure below.

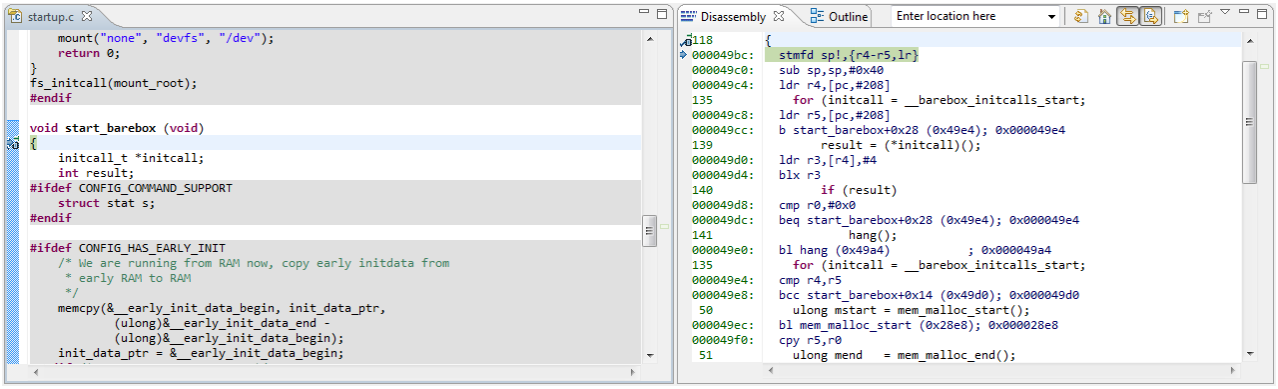
Figure 24. Set a hardware breakpoint at start_barebox



7. Resume debugging using **F8** or Debugger Shell command *`go`*.

   Breakpoint will be hit and debugging (step, run, or breakpoint) can be done until the Barebox boots up.

Figure 25. Perform debugging



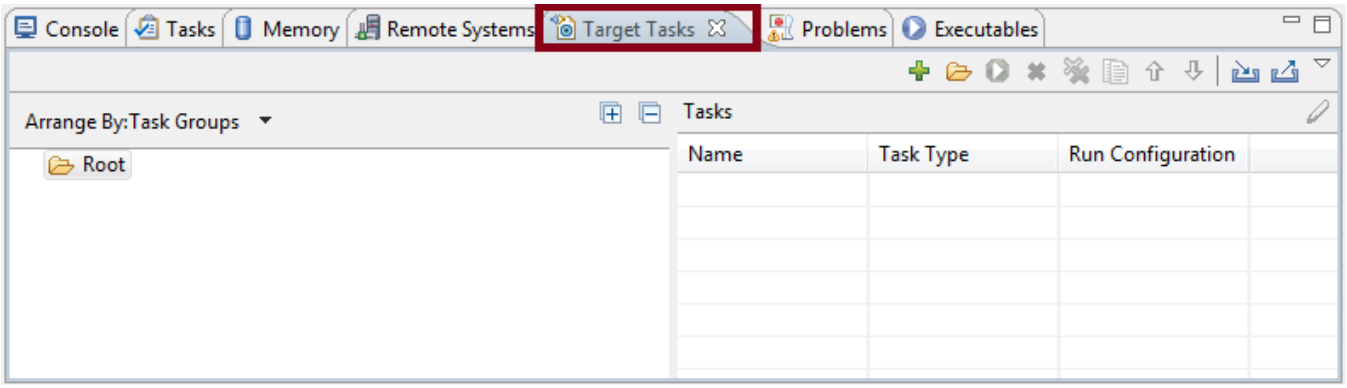
7. Downloading U-Boot/Barebox binary on target board

U-Boot/Barebox binary must be downloaded into the flash device on the target board, and must be from the same build as the U-Boot/Barebox image that is imported as a CodeWarrior project.

Perform these steps to download the U-Boot binary on the target board (the steps to download Barebox binary are similar, only the addresses may differ):

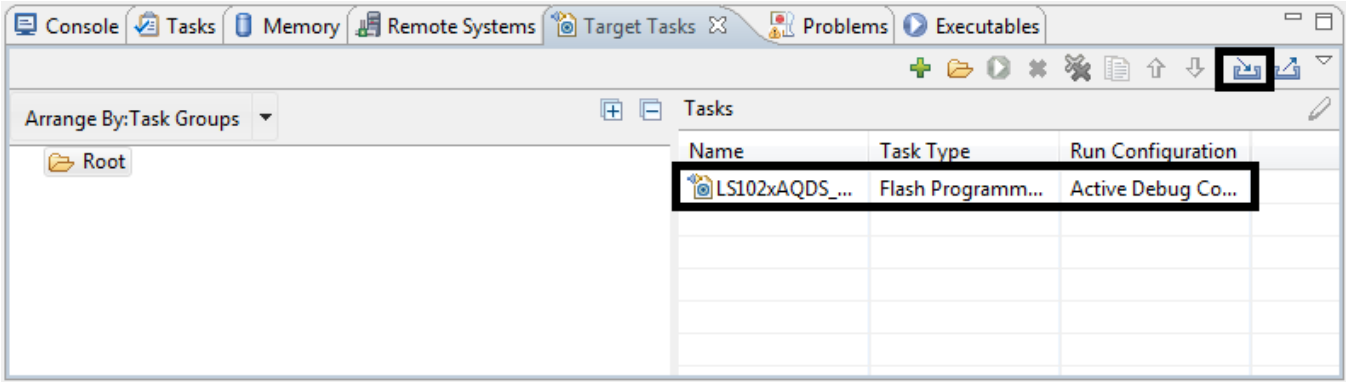
1. With the board in debug, open the **Target Tasks** view.

Figure 26. Target Tasks view



2. Click the **Import** icon, and import the target task (see figure below).

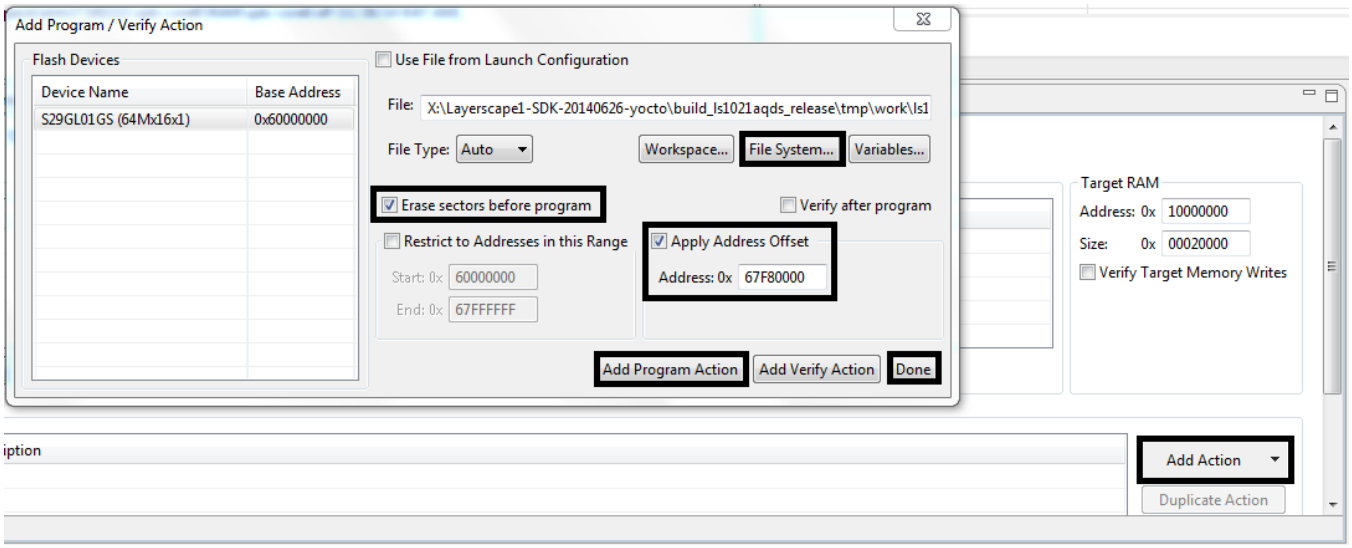
Figure 27. Import target task



3. Open the imported target task in the **ARM Flash Programmer Task** window.
4. Click the **Add Action** down arrow and choose **Program / Verify**.

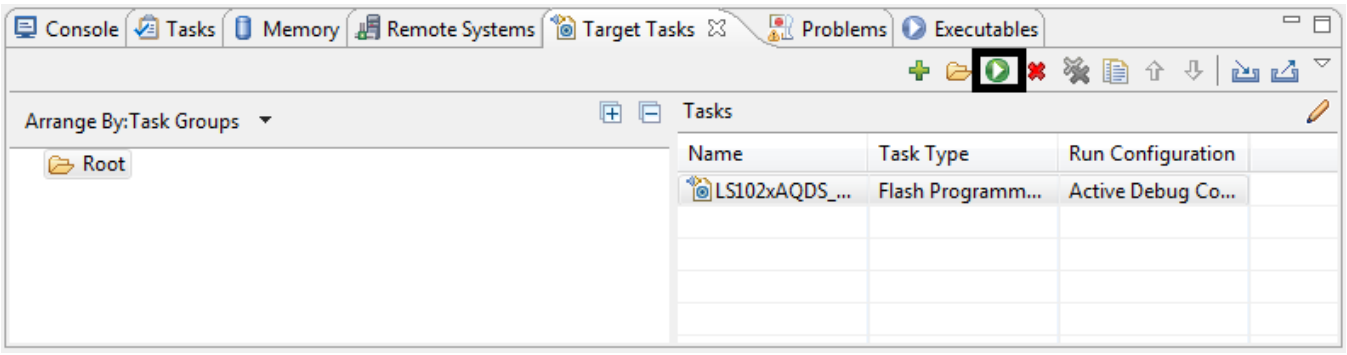
   The **Add Program / Verify Action** dialog opens.
5. Browse for the U-Boot binary. In addition, ensure that the **Erase sectors before program** and **Apply Address Offset** checkboxes are selected, and correct address is specified for the **Apply Address Offset** option.
6. Click **Add Program Action**, and then click **Done**.

Figure 28. Add Program / Verify Action dialog



7. Execute the target task, as shown in the figure below.

Figure 29. Execute target task



8. After the downloading is complete on the target board, the U-Boot binary will be available that is in sync with the U-Boot image.

NOTE	The example was done for downloading U-Boot binary to the NOR flash. If the NAND flash is used, ensure that correct target task address is specified for the flash device, specific U-Boot binary is used, and correct offset address is set. For more details about Flash Programmer, see Chapter 11 of *CodeWarrior Development Studio for QorIQ LS series - ARM V7 ISA Targeting Manual*.

8. Calculating PIC load address for U-Boot DDR relocation

To calculate the new PIC load address, after U-Boot relocation, apply this formula:

```
PIC address = Runtime symbol address (RAM symbol address in our case)
- Compile time symbol address
```

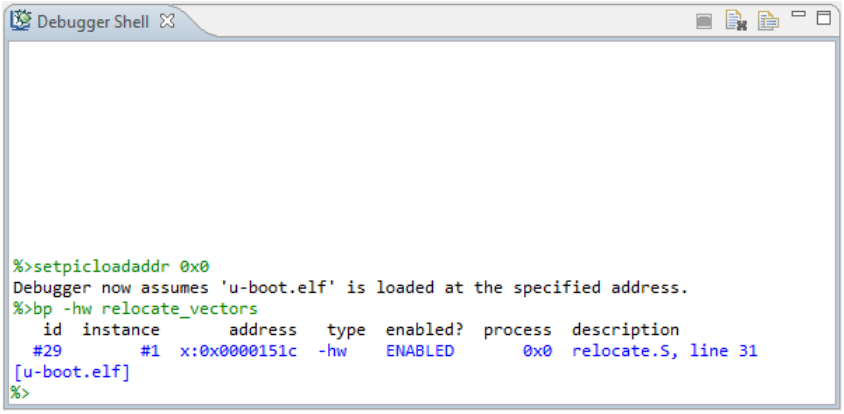
After step into `bx lr`, in Debugger Shell, perform these operations:

1. Set PIC load address to `0x0`, using Debugger Shell command `setpicloadaddr 0x0`. It tells the debugger that the main executables are loaded at `0x0`.

NOTE	This is not the same as `setpicloadaddr reset` command, which tells the debugger that the main executables are loaded at the address set in the ELF file.

2. Set a hardware breakpoint at the function code will jump to. In our case, this is `relocate_vectors`. It shows the compile-time symbol address.

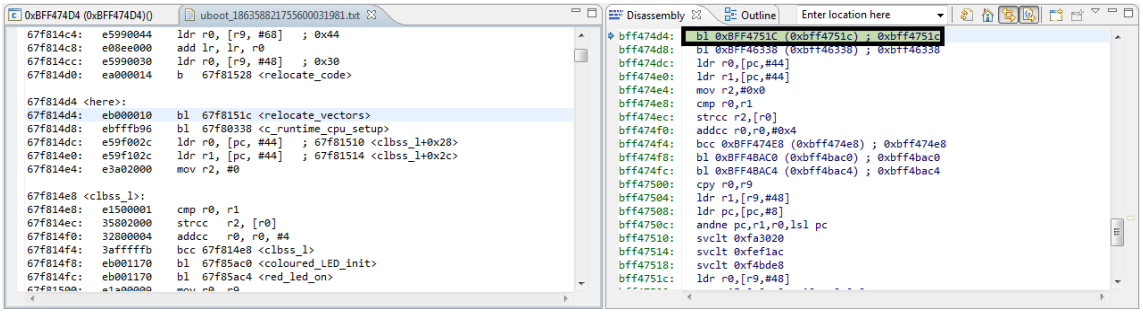
Figure 30. Set a hardware breakpoint



3. Calculate the difference between the runtime symbol address (single step after *bx lr* instruction, using the address *relocate_vectors* will jump to) and compile-time symbol address:

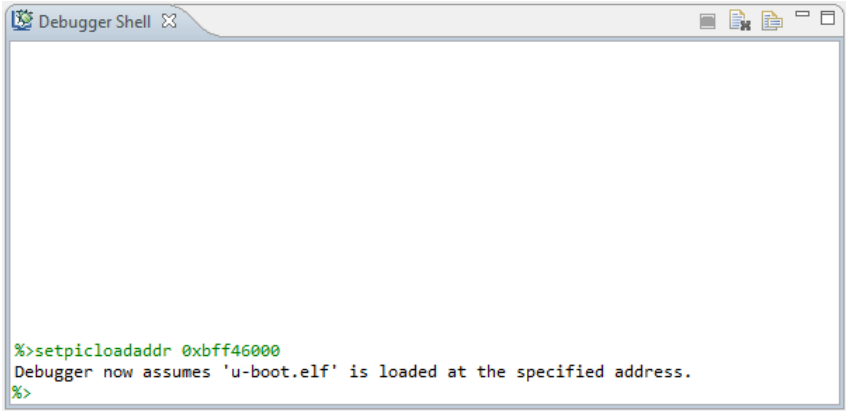
   PIC address = 0xBFF4751c (relocate_vectors jump address) – 0x0000151c (relocate_vectors breakpoint address) = 0xBFF46000

Figure 31. Calculate PIC load address



5. Set the PIC load address to *0xBFF46000* using the Debugger Shell command *setpicloadaddr 0xBFF46000*.

Figure 32. Set PIC load address



How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

Document Number: AN4956

7 July 2015