Freescale Semiconductor
Application Note

Document Number: AN5065

Adding Custom Registers to the CodeWarrior for Power Architecture® Processors

Contents

1. Introduction

This document describes how to add new registers to the debugger database of the CodeWarrior Development Studio for Power Architecture® Processors that can be viewed in the **Registers** view of the CodeWarrior IDE.

CAUTION Modifications to the debugger database files that are invalid may corrupt the CodeWarrior for Power Architecture installation on your machine. Restoring the original database files should be sufficient to recover from any problems. If not, a complete re-installation of the CodeWarrior for Power Architecture software is required.

This document helps you to create new register files and package them in a database file that will be loaded and interpreted by the debugger.

NOTE Any modifications made to the CodeWarrior software, files, and folders as described in this application note are not supported by default and should be made at your own risk.

2. Background

The debugger uses database files for storing persistence information, such as register information. A database file is actually a collection of archived XML files. It has the `.mwpdb` extension. Each processor type has a separate database file.

A register is defined in a register-definition file, which is an XML file that provides detailed information (such as the meaning of each bit in the register) about a register.

For each processor, the register-definition files of all the registers that belong to the processor are bundled in a .zip file that is named for the processor. Then, the file extension of each ZIP file is changed from `.zip` to `.mwpdb`. For example, the `.mwpdb` file that contains the register-definition files of all the registers that belong to the QorIQ T1040 processor is `PowerPC_EABI_T1040.mwpdb`.

Each processor has the following two types of registers:

- Processor-specific registers: Includes register set that is specific to the processor.
- Core-specific registers: Includes register set that is specific to the core used in the register. Registers specific to a core are common for all processors that use the core.

3. Finding and extracting database files

For the CodeWarrior for Power Architecture v10.x debugger product, the database files are located in:

- For Windows:

  ```
  <CWInstallDir>\PA\bin\plugins\support\Products\ProductData
  ```

  where `<CWInstallDir>` is `C:\Freescale\CW_PA_v10.x.x` in a default installation.

- For Linux:

  ```
  <CWInstallDir>/PA/CodeWarrior/CodeWarrior_Plugins/Support/Products/ProductData
  ```

  where `<CWInstallDir>` is `/opt/Freescale/CodeWarrior_PA_v10.x.x` in a default installation.

Perform the following steps to find and extract the correct database file:

1. Locate the appropriate grouping (`<product-name>.mwpdb` file) of register-definition files. For Windows, go to the following folder:

```
<CWInstallDir>\PA\bin\plugins\support\Products\ProductData
```

For Linux, go to the following folder:

```
<CWInstallDir>/PA/CodeWarrior/CodeWarrior_Plugins/Support/Products/ProductData
```

2. Make a backup copy of the `<product-name>.mwpdb` file and save it in another folder.
3. Change the file extension of the `<product-name>.mwpdb` file to `<product-name>.zip`.
4. Using a standard zip utility, unzip the `<product-name>.zip` file to the following folder (for Windows):

   ```
   <CWInstallDir>\PA\bin\plugins\support\Products\ProductData\<product-name>.mwpdb
   ```

   or to the following folder (for Linux):

   ```
   <CWInstallDir>/PA/CodeWarrior/CodeWarrior_Plugins/Support/Products/ProductData/<pro
   duct-name>.mwpdb
   ```

5. Delete the `<product-name>.zip` file.

When finished, the `ProductData` folder should look like it did before this process began, except that now `<product-name>.mwpdb` will be a folder, not a file. The `<product-name>.mwpdb` folder contains register-definition files and these files are used by the CodeWarrior debugger in their unzipped form in the same way that they are used in their original zipped form. You can open the `<product-name>.mwpdb` folder and see the structure of its files and folders.

In this example for T1040, we end up with the file `PowerPC_EABI_T1040.zip` (now deleted) and the folder `PowerPC_EABI_T1040.mwpdb`.

4. Adding custom definitions

Once the register-definition files are extracted, you need to edit them to add in the new or custom register definitions. In this example, which is performed on Windows, 13 BMAN (DPAA) registers are added to the list of T1040 registers.

The `.zip` file associated with this Application Note contains the .xml files described in the following steps, which are intended to be used as references.

NOTE The file `PowerPC_EABI_T1040.mwpdb` is available at: PowerPC_EABI_T1040.mwpdb.zip. This file contains all the XML files and folders that are either modified or created while going through the procedures described in this Application Note.

Perform the following steps to add custom definitions:

To add one or more new registers, start by editing the `product-manifest.xml` file, which is located in the `<CWInstallDir>\PA\bin\plugins\support\Products\ProductData\PowerPC_EABI_T1040.mwpdb`.

NOTE You can edit these XML files in any text editor, though an XML editor is recommended.

At the top of the `product-manifest.xml` file, locate the XML header `<register-details>`. This group is an alphabetical list of all the individual registers that are displayed in the CodeWarrior for Power Architecture Debugger's **Registers** view. Since BMAN is the new first entry in this group, add the following lines above the `CCM_CCLR0` entry, after `<register-details>`:

```
 <file>
  <name>BMAN_BCSP0_CR</name>
  <version>1.0</version>
  <path>RegisterDetails/QorIQ/T1040/BMAN/BMAN_BCSP0_CR.xml</path>
</file>
```

The referenced `BMAN_BCSP0_CR.xml` file is created later and provides the register details for the `BMAN_BCSP0_CR` register.

1. Repeat Step 1, for every BMAN register to be added to the CodeWarrior for Power Architecture Debugger's **Registers** view. This example adds only 13 registers: `BMAN_BCSP0_CR`, `BMAN_BCSP0_RR0`, `BMAN_BCSP0_RR1`, `BMAN_BCSP0_RCRn (0-7)`, `BMAN_BCSP0_RCR_PI_CENA`, and `BMAN_BCSP0_RCR_CI_CENA`.
2. Find the XML header `<register-collection>` in the `product-manifest.xml file`. This header starts the ordered list of register groups in the T1040, organized by each group's base address in the T1040 memory map. According to the *QorIQT1040 Reference Manual*, the BMAN block's base address is `31_A000h`, and the register group (that is already displayed in **Register Details** view) just above it is Security Monitor, at `31_4000h`.
3. Below the `<register-collection>` XML header, find `SECMON (SECMON)`, which looks like this:

```
<file>
  <name>SECMON (SECMON)</name>
  <version>1.0</version>
  <path>Registers/QorIQ/T1040/SECMON/SECMON.xml</path>
</file>
```

Insert the following lines after the `</file>` line of the `SECMON` entry:

```
<file>
  <name>BMAN (BMAN)</name>
  <version>1.0</version>
  <path>Registers/QorIQ/T1040/BMAN/BMAN.xml</path>
</file>
```

The referenced `BMAN.xml` file will be created in Step 6 below and will provide the register mapping for all the newly-added BMAN registers.

4. Create the folder `<path>PowerPC_EABI_T1040.mwpdb\Registers\QorIQ\T1040\BMAN`.

5. In the new `BMAN` folder, create a blank file `BMAN.xml`. Using an existing file such as `SECMON.xml` as a guide, fill in the contents of `BMAN.xml` with header information and entries for each of the 13 BMAN registers defined in Step 1. See Table 4-1, BMAN Software Portal Memory Map, of the *T1040 QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* for address offsets for each register.

6. Create a folder `<path>PowerPC_EABI_T1040.mwpdb\RegisterDetails\QorIQ\T1040\BMAN.`

7. In the new BMAN folder, create the empty file `BMAN_BCSP0_CR.xml`. Using an existing file such as `SECMON_HPLR.xml` as a guide, fill in the contents of `BMAN_BCSP0_CR.xml` with details for each bit field, as defined in the *T1040 QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual*.

   This step takes the most work, defining all the `bitfields` for every new register to be added to the debugger's **Registers** view. A convenient temporary shortcut is to use the following simplified register-details example instead:

```
<register-details-file>
  <register-details>
  <version>0.1</version>
    <name>BMAN_BCSP0_CR</name>
    <bitrange>0:63</bitrange>
    <reset-value>0x00000000</reset-value>
    <description>BMan software portal 0, command register.</description>
    <bitfields>
      <bitfield>
        <name>-</name>
        <bitrange>0:63</bitrange>
        <format>binary</format>
        <access>readwrite</access>
        <description>BMan software portal 0, command register.</description>
      </bitfield>
    </bitfields>
  </register-details>
</register-details-file>
```

   Change the `<name>`, `<bitrange>`, `<reset-value>`, `<description>`, and `<access>` fields as appropriate for each new register.

8. Edit the file `<path>Registers\QorIQ\T1040\DefaultLayout_cpuPowerPCBig.xml` and add the following line immediately after the entry for `/SECMON/SECMON.xml`:

```
<register-file-name>Registers/QorIQ/T1040/BMAN/BMAN.xml</register-file-name>
```

   The `DefaultLayout_cpuPowerPCBig.xml` file determines the order in which the register groups are displayed in the debugger's **Registers** and **Register Details** views. The BMAN block base address is `31_A000h` and the Security Monitor block base address is `31_4000h`, so the BMAN registers would logically appear immediately after SECMON registers.

9. When all the new registers have been defined, delete the file `<CW_PA10>\PA\bin\plugins\support\Products\ProductData\chameleon_toc.sqlite`, and then launch or restart the CodeWarrior for Power Architecture IDE. It will take a few moments to process the newly organized registers information. You will see the console status message, **Indexing debug database**, while doing so.

10. After all the background processing is completed, open the **Register Details** view, select the processor for which the new registers were added, and scroll down the list of register groups until the new group is found. Expanding the new group shows all the individual registers in that group, and clicking on any register shows the register details. New registers and register groups can be added incrementally, as needed.

11. When all changes and additions are made and verified, create a `.zip` of all the files in the `PowerPC_EABI_T1040.mwpdb` folder (you can either move all the original, new, and modified files and folders into the `.zip` file, or delete the files and folders after copying them to the .zip file), then change the extension from `.zip` to `.mwpdb`.

12. Delete `chameleon_toc.sqlite` one more time and launch or restart the CodeWarrior for Power Architecture to verify that all registers are displayed as expected.

Document Number: AN5065

15 July 2015