Freescale Semiconductor
Application Note

Document Number: AN4502
Rev. 0, 2012

RAppID Source Code Integration using CodeWarrior 10.2 for the PX Family

by: David Connelly
IMM Applications

Contents

1 Introduction

This document outlines the procedure for integrating RAppID-generated source code into a stationery-based CodeWarrior 10.x project for the PX family of devices. The RAppID tool creates a full library of source code including startup files (that is, crt0.s), interrupt handling, and peripheral initialization source code that can be used with several types of compilers—CodeWarrior, Wind River, or Green Hills. This document details the steps necessary for integrating the full library of assembly and C-based source code into the CodeWarrior 10.x environment. This particular example is specific to the PXS30 (DPM mode) and PXS20 (LSM mode) dual-core architectures, but most steps apply to any PX family derivative.

2 Integration

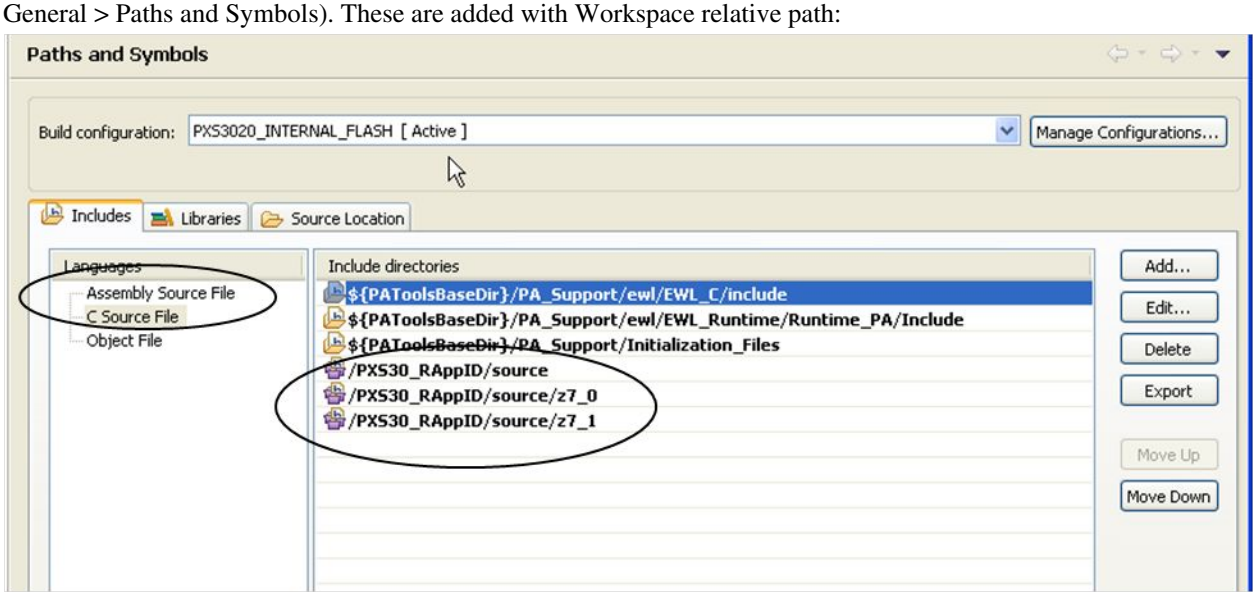
1. **Create base project within CodeWarrior 10.x.** Create new project from stationery (File > New > Project … > Bareboard Project). For the PXS30 example, VLE code format was chosen.
2. **Update directory structure.** Remove all source code folders and contents. Right-click, Delete. For example, PXS30 stationery includes the following folders:
   - PRC0_Sources
   - PRC1_Sources

- Project_Headers
- Sources
- Startup_Code—subfolder within Project_Settings. (Keep Debugger and Linker_Files folders.)

NOTE

Keep the Prefix folder for now (this will be discussed later). Folders containing target names such as PXS3020_INTERNAL_FLASH can be kept since these simply contain output files. The Project_Settings\Debugger folder contains startup and initialization scripts specific to the debugger, and Project_Settings\Linker will hold the linker files, so these both need to remain in the project.

3. **Plug in RAppID files.** (Drag and drop or Import)
   - You will first need to install and run both the RAppID pin wizard and init tool to create the source code. Alternatively, example source code may be available on the device website (for example, http://www.freescale.com/twr-pxs20).
   - When importing files, CodeWarrior Eclipse will ask to link the file or to make a copy. For this example, choose to make a copy since we want all the source code to be linked relative to the main project path (D:\CodeWarrior_PX for example). Do not link the files from the RAppID source directory (D:\Rappid\...).
   - Right-click on the newly imported source folder and go to Resource Configurations > Exclude From Build. Make sure files are included in both targets, RAM and flash (when box is checked it will exclude it from that build, so uncheck box to include files in that target). For more detail on possible differences in RAM vs flash source code differences, skip to step 9.
   - Delete the following files from newly imported src folder, as they are not part of the source code build (right-click and delete):
     - Makefile
     - *.bat files
   - Move the .lcf file that was created with the RAppID tool into the Linker Files folder in the CodeWarrior project path. Please note, the RAppID initialization tool will generate linker files particular to the target configuration (RAM or flash memory), so be sure to include the proper file into the correct target (RAM or flash memory). After it is included in the CodeWarrior project, be sure to exclude the RAM linker file from flash target, and vice versa. After including the RAppID linker files, the original CodeWarrior linker files can be deleted. More detail on linker file importing can be found in Step 15.
4. **Update include directories**. Add the include directories in the project settings so header files can be found during the build. This has to be done for assembly and C source files—note both tabs in the example below. The RAppID-generated source code formatting creates header files and source files and puts them in the same folders. In other words, there isn't a specified folder for header files. This can be changed later, but for simplicity during the initial integration process we will keep the same directory structure as the RAppID tool. Example (Properties > C/C++ General > Paths and Symbols). These are added with Workspace relative path:



5. **Check header file.** Make sure the header file names within the source code match the header file name generated with the RAppID tool. For example, PXS3020.h.
6. **Disable C++ exceptions.** Project Properties > Settings > C/C++ Language > Uncheck “Enable C++ Exceptions.”
7. **Check coding errors.** If necessary, fix any syntax errors from the RAppID tool—for example, your specific interrupt handler code, which can be input through the wizard and which cannot be checked for syntax errors until code is generated and compiled.
8. **Check Prefix file.** RAppID source code generates a prefix file (for example, vle_macro.txt in the src folder) and CodeWarrior also creates one with project stationery (for example, PXS3020_INT_FLASH_VLE_DPM.prefix, in the Prefix folder). Choose the CodeWarrior stationery prefix file to include in the build, as included paths are already set up to use this file. Before the RAppID prefix is deleted, do a “diff” on both to make sure the proper definitions exist for the build. For example, ensure that the prefix file has VLE_IS_ON defined (only if the VLE option is selected in the project stationery AND the RAppID code generation utility). As mentioned, the prefix file is in a folder named “Prefix” that is created with the project stationery. Make sure project points to the correct prefix file by going to Properties > Settings > PowerPC Compiler > Preprocessor.

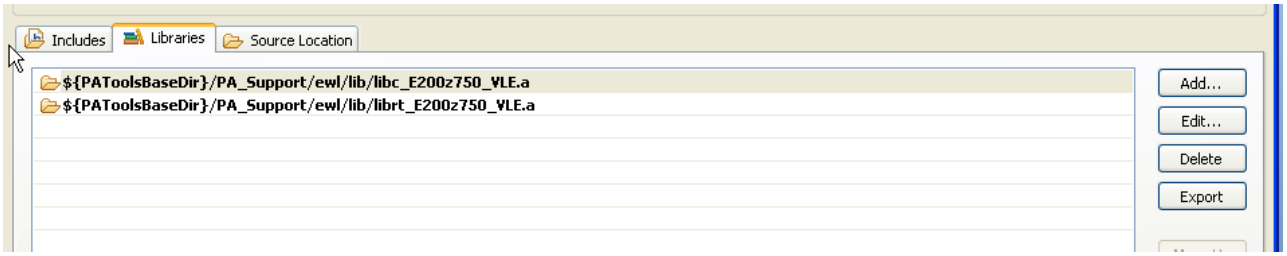
NOTE

If no prefix file exists, create a header file (File > New > Header file) named prefix.h and include it in project source directory. Delete the original prefix folder if it exists. Add “#define VLE_IS_ON 1” if using VLE code. Include any other definitions required for the build, and add this file to the preprocessor settings by going to Properties > Settings > PowerPC Compiler > Preprocessor. This prevents having to do a “#include prefix.h” everywhere in the source code.

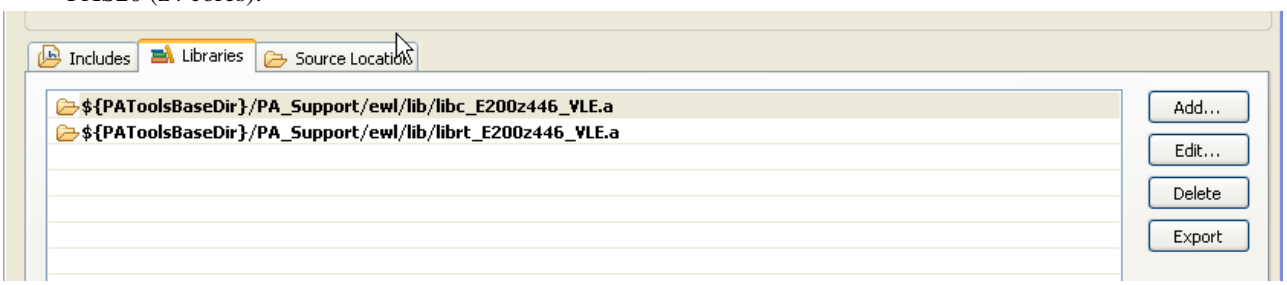
9. **Support flash and RAM targets with RAppID-generated code**. RAppID-generated source code can be different depending on RAM or flash target configuration. This is particularly evident with PXS30 in DPM mode, since the build configuration changes considerably when going from a flash target to a RAM target. This is because each half of the RAM block becomes dedicated to one core for code execution. To get one set of source code files to support both RAM and flash targets, a “diff” may need to be performed on the source code generated with RAppID tool to understand where compile time support needs to be added.
    - For the PXS20 in LSM mode (or devices with only one core), you may only need to import different startup files (crt0_p0.s) for both RAM and flash targets. If it is not necessary to support both targets, then import the RAppID source code built for either RAM or flash, and import only one linker file. This results in a project with only one target.
    - Below is an example of a .c file from the z7_1 folder for PXS30 (dual core device in DPM mode) that needs compile time support, as seen here in sys_init_fnc_p1.c:

```
#if RAM_TARGET
#pragma section code_type ".text_vle1"
__declspec (section".text_vle1")    void sys_init_fnc_p1(void);
#endif
```

    - Note that RAM_TARGET or FLASH_TARGET symbols can be defined in the prefix file. For example, in the flash prefix file: #define FLASH_TARGET 1 and #undef RAM_TARGET. For the RAM prefix file: #define RAM_TARGET 1 and #undef FLASH_TARGET.
    - For the RAM target, right-click on file rchw_init.c and exclude it from the build. This is the reset configuration half-word that is put into flash memory for the flash target only.
10. **Update assembler prefix symbols.** The assembler also needs to use the same symbols that are in the prefix file. Go to Properties > Settings > PowerPC Assembler > General.
    - In the Other Flags box, enter “-define VLE_IS_ON.” This will allow the symbol VLE_IS_ON to be recognized in all the assembly files. If it is not, several errors will be reported since the non-VLE assembler syntax is different than VLE assembly.
11. **Remove library files.** Go to Properties > C/C++ General > Paths and Symbols > Libraries (tab) and delete the ".a" library files. This prevents conflicts with the __start entry point, which will typically point to startup code within the runtime library shown below.
    - PXS30 (z7 cores):

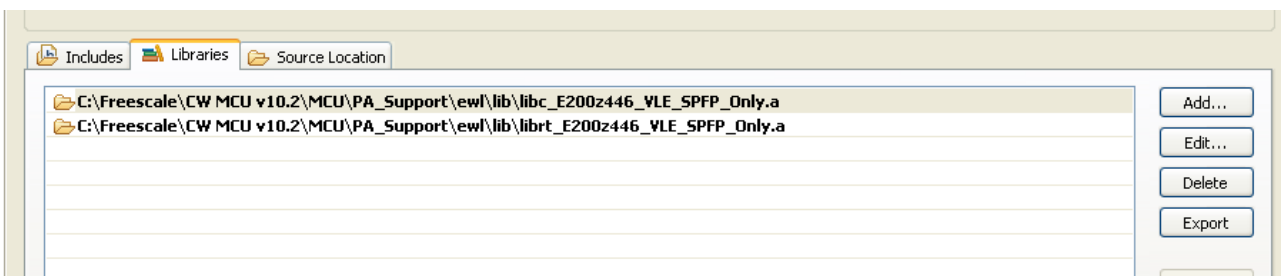


- PXS20 (z4 cores):

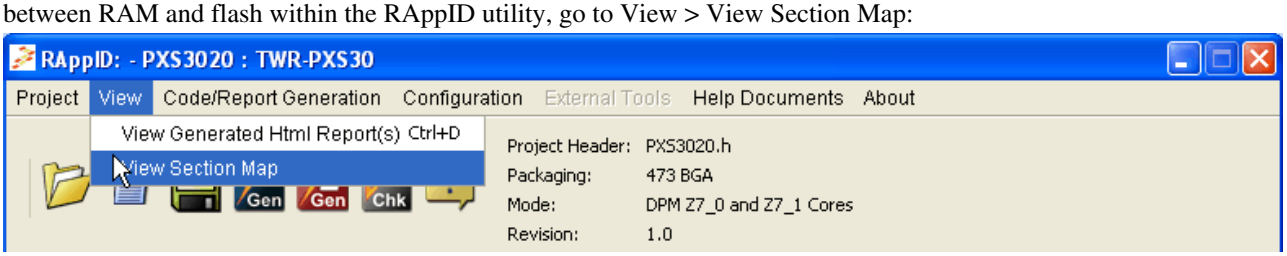


NOTE

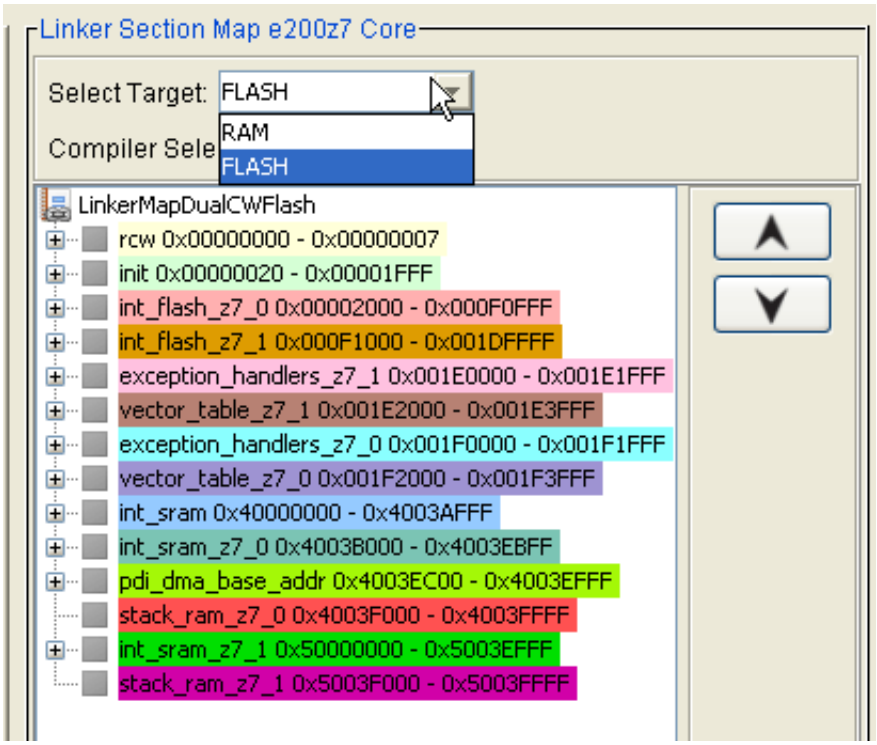
If you need floating point support and do not want to include full runtime support, the floating point "only" libraries can be imported, as shown below.



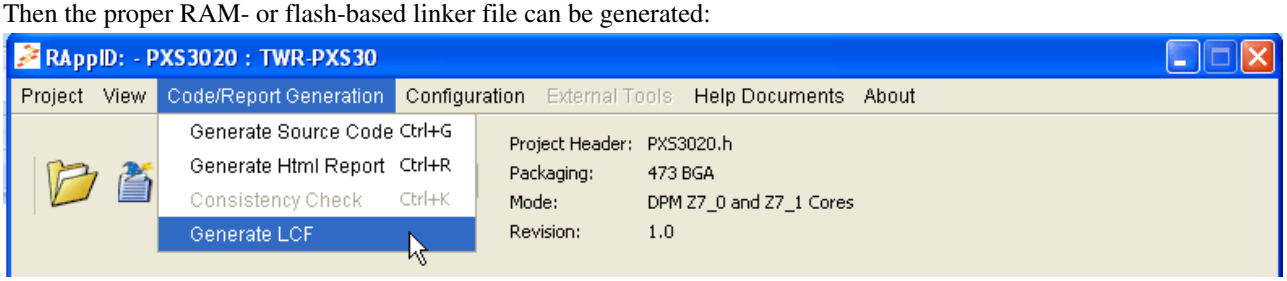
12. For flash target, **uncheck box "Generate ROM Image."** Go to Properties > Settings > PowerPC Linker > Output tab.
13. If you are using a standalone printf file, **select "Make Strings Read Only"** in Properties > PowerPC Compiler > Processor. Otherwise, the string will not be recognized.
14. **Change the program's entry point**. Update both targets if they exist (for example, RAM and flash). Go to Properties > PowerPC Linker > Input, and change Entry Point from __startup to __start.
15. **Delete CodeWarrior linker files and update with RAppID linker files** (initially discussed in step 3). The RAppID linker files are generated and put into the default source output folder based on the RAM or flash target. To select between RAM and flash within the RAppID utility, go to View > View Section Map:



The pulldown menu, seen below, selects between RAM and flash:



Then the proper RAM- or flash-based linker file can be generated:



To generate BOTH linker and source files, use the Generate Source Code option from the pulldown menu. Prior to doing this, you should specify different directories for the RAM- and flash-generated source code.

When the linker files have been generated, import each one into the CodeWarrior project for the appropriate target. Then, update the settings so each target uses the correct linker command file by going to Properties > Settings > PowerPC Linker > Input > Link Command File. Click "Browse…" and choose the new linker files. This process needs to be done for both RAM and flash targets.

NOTE

You may encounter compiler warnings for missing function prototypes. These are easily resolved by adding prototypes to the appropriate header file.

3 Conclusion

The RAppID initialization and pin wizard tool is a powerful way to create a complete library of customized source code, as well as full documentation to support all the details. The CodeWarrior 10.x Eclipse environment allows easy integration of source files, flexible build options, and powerful debug environment for any user. Compile, debug, and enjoy your new project!

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor
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: AN4502
Rev. 0, 2012

freescale semiconductor