Lab 1 Getting Start with ADS

Table of Contents

IP Core Design
2002.9

1. Creating an Application

This Chapter describes how to create an application using ADS. The contents are based on the Chapter 3 of "ARM Developer Suite Version 1.1Getting Started Guide". See the **CodeWarrior IDE Guide** for more information on what is not described in this document or "ARM Developer Suite Version 1.1Getting Started Guide".

To let CodeWarrior IDE display menu normally, font size of Microsoft Window should be set as **small font**.

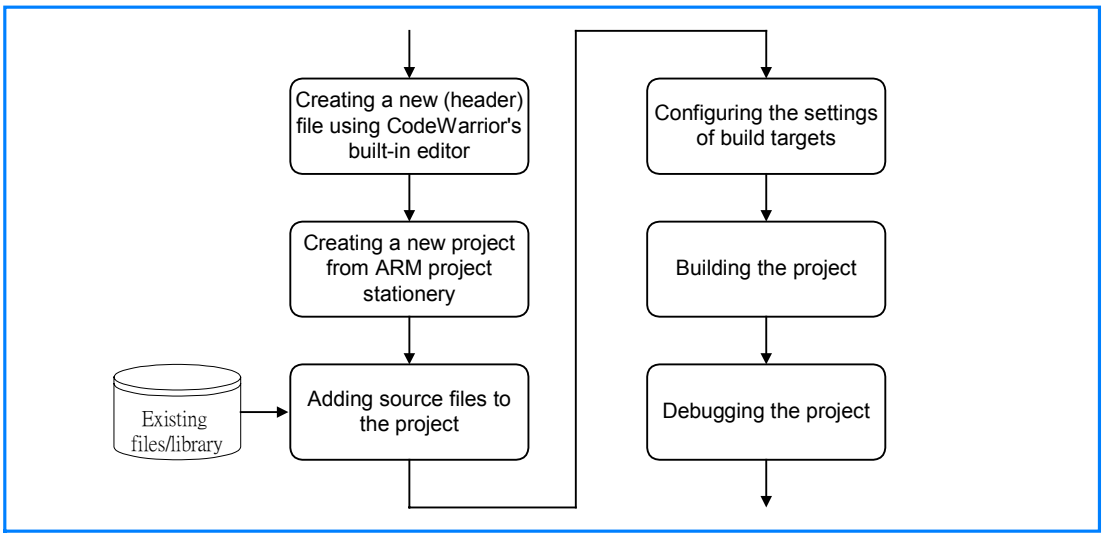

Figure 1. Flow diagram of Lab 1.

1.1 Using the CodeWarrior IDE

The CodeWarrior IDE provides a graphical user interface to configure the ARM tools to compile, assemble, and link your project code. It enables you to organize source code files, library files, other files, and configuration settings into a ***project***. Each project enables you to create and manage multiple ***build targets***. A build target is the collection of build settings and files that determines the output, which is created when you build your project. Build targets can share files in the same project, while using their own build settings.

CodeWarrior for the ARM Developer Suite provides preconfigured project stationery files for common project types, including:

- ARM Executable Image
- ARM Object Library
- Thumb Executable Image
- Thumb Object Library
- Thumb/ARM Interworking Image.

You can use the project stationery as a template when you create your own projects. The non-interworking ARM project stationery files define three build targets. The interworking (i.e., Thumb/ARM Interworking Image) project stationery defines an additional three build targets to compile Thumb-targeted code. The basic build targets

for each of the stationery projects are:

- **Debug**: This build target is configured to build output binaries that are fully debuggable, at the expense of optimization.
- **Release**: This build target is configured to build output binaries that are fully optimized, at the expense of debug information (i.e., no source level debug information, but full optimization).
- **DebugRel**: This build target is configured to build output binaries that provide adequate optimization, and give a good debug view. This is a trade-off between Debug and Release.

1.1.0 Creating a new header file using CodeWarrior's built-in editor

1. Select **Programs → ARM Developer Suite → CodeWarrior for ARM Developer Suite** from the Windows **Start** menu to start the CodeWarrior IDE.
2. Select **File → New Text File (or Ctrl+N)**.
3. Enter the following C text. Make sure that "#defin" instead of "#define" is typed.

```
1  /* This preprocessor results in the C library function clock () begin used for timing
2  measurements.*/
3
4  #defin MSC_CLOCK
```

Figure 2. Simple C header file.

4. Select **File → Save As (or Ctrl+N)**.
   4.1 Navigate the directory structure to c:\.
   4.2 Create new directory c:\ARMSoC\Lab_01\.
   4.3 And ether the filename dhry_def.h
5. Click **Save**. Click **Yes** to overwrite (if necessary).

1.1.1 Creating a new project from ARM project stationery

1. Select **File →New… (or Ctrl+Shift+N)**. A New dialog is displayed (Figure 3).
2. Ensure that the **Project** tab in Figure 3 is selected. The available ARM project stationery is listed in the left of the dialog, together with the Empty Project stationery and the Makefile Importer Wizard.

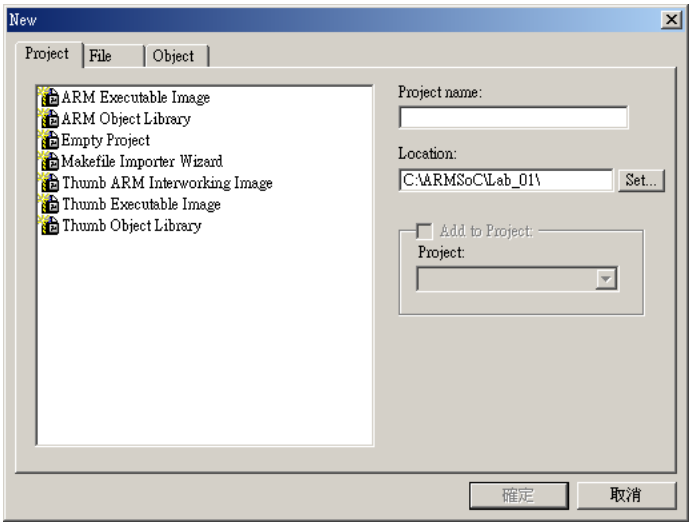

Figure 3. New dialog.

3. Select **ARM Executable Image** from the list of project stationery.
4. Set the directory where you want to save the project in the **Location** field or click the **Set…** button (a **Create New Project** dialog is displayed) next to the **Location** field to navigate to the directory c:\ARMSoC\Lab_01\.
   Enter a project name, for example My_Project. A result is shown in Figure 4.

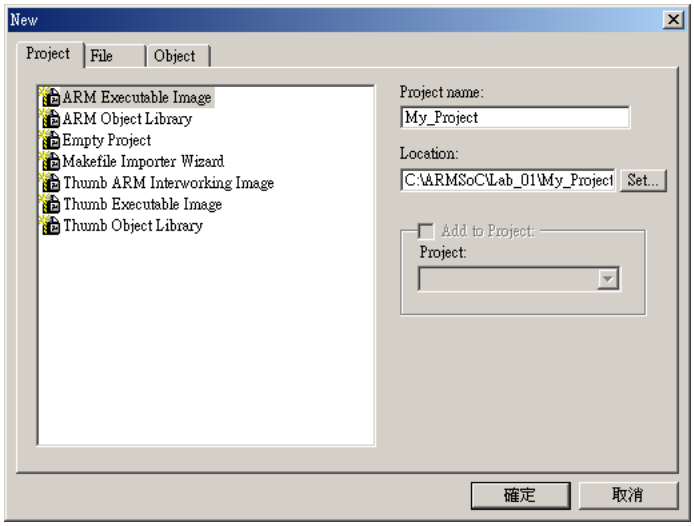

Figure 4. Setting project name and location path.

5. Click **OK**. The CodeWarrior IDE creates a new project based on the ARM Executable Image project stationery, and displays a new project window with the **Files** tab is highlighted and DebugRel is selected as the build target by default (Figure 5). Other build targets can be selected by clicking on the drop-down box. It is the DebugRel variant that we shall use for the remainder of this Lab.

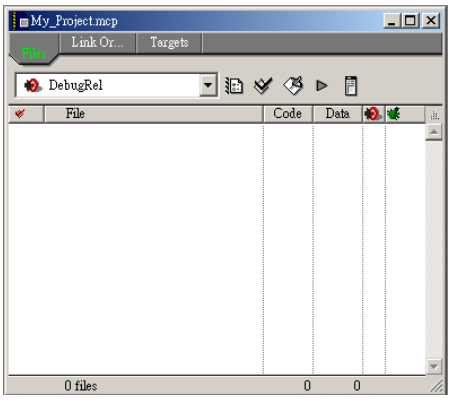

Figure 5. New Project.

After the last step, the following directories and files are created.
PATH\My_project\My_project.mcp
PATH\My_project\My_project_Data\CWSettingsWindows.stg
PATH\My_project\My_project_Data\Debug\TargetDataWindows.tdt
PATH\My_project\My_project_Data\DebugRel\TargetDataWindows.tdt
PATH\My_project\My_project_Data\Release\TargetDataWindows.tdt

- Close ADS
- Double click on My_project.mcp, the ADS IDE starts and displays the project window as it did at step Five (Figure 5).

1.1.2 Adding source files to the project

1. Ensure that the **project** (titled as My_Project.mcp in this example) window is the active window.
2. Select **Project →Add Files…**(Figure 6). A **Select files to add…** dialog is displayed. Navigate to the **dhryansi** directory in the **install_directory\Examples** (e.g., C:\Program Files\ARM\ADSv1_1\Examples\dhryansi\) directory and Shift-click on **dhry_1.c** and **dhry_2.c** to select them (Figure 7).

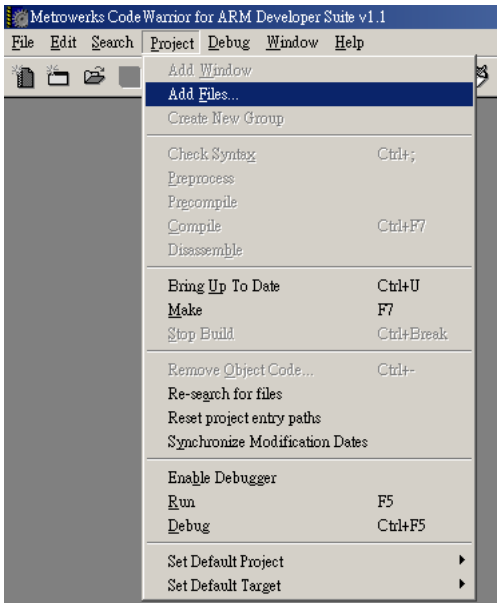

Figure 6. Add file.

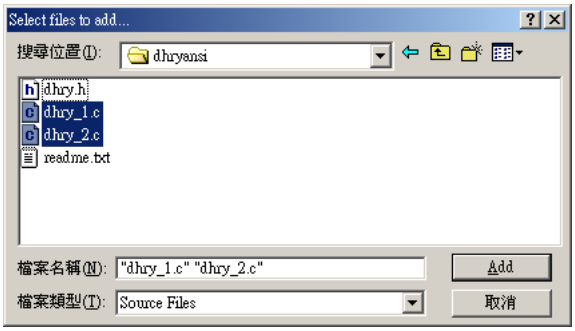

Figure 7. Select files to add… dialog.

3. Click **Add**. The CodeWarrior IDE displays an **Add Files** dialog (Figure 8). The dialog contains a checkbox for each build target defined in the current project. In this example, the dialog contains three checkboxes corresponding to the three build targets defined in the ARM Executable Image project stationery.

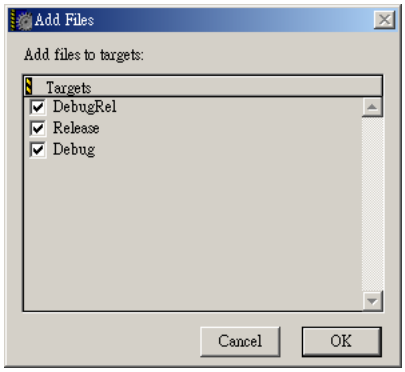

Figure 8. Add Files.

4. Leave all the build target checkboxes selected and click OK. The CodeWarrior IDE adds the source files to each target in the project and displays a Project Messages window (Figure 9) to inform you that the directory containing the source files has been added to the access paths for each build target.

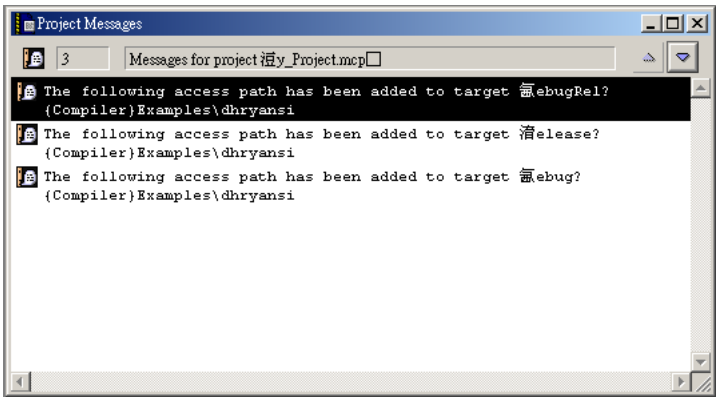

Figure 9. Project Message window.

Note: The access paths for each build target define the directories that will be searched for source and header files. You do not need to explicitly add the header files for the dhryansi project because the CodeWarrior IDE locates them in the newly added access path. As the message "The following access path has been added to target "DebugRel": {Compiler}Examples\dhryansi" shown in the Project Message window. However, you can add header files explicitly if you want, follow the instruction described in step 2 of Section 1.1.3.

Repeat step 2~4 to add the dhry_def.h file you build in Section 1.1.0.

5. Ensure that the Files tab is selected in the project window. The project window displays all the source files in the project. (Figure 10).

Build Target pop-up menu

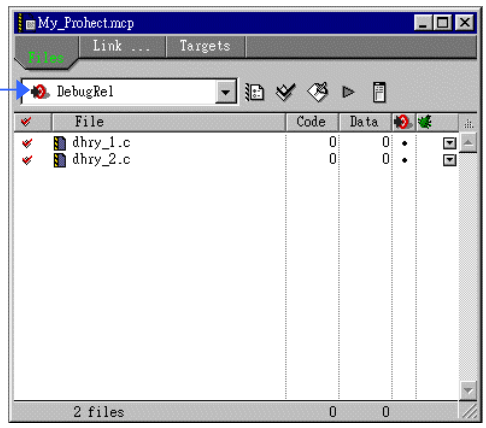

Figure 10. Source files in Files view.

6. Select dhry_def.h from the project window and click right button on it. Select Preprocess. After preprocessing this header file, two windows appear. At Error and Warring window, two meeafes are sown as below:

Error : (Serious) C2858E: Unknown directive: #defin
dhry_def.h line 2

C:\ARMSoC\Lab_01\dhry_def.h: 0 warnings, 0 errors, 1 serious error

The Lower section of the window contains a section of the code that caused the first error message.

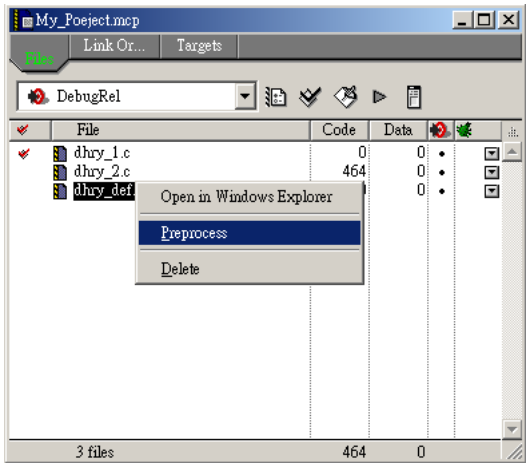

Figure 11. File handling in Project Window.

Note:

- Source files in the project window can be edited by double clicking on their icons.
- Select the file in the project window and press “delete” will remove the file added in step 2~4 form the project.
- If you close CodeWarrior IDE after the fifth step, the files added to My_project.mcp are automatically saved without your indication. Double click on My_project.mcp, the CodeWarrior IDE starts and displays the project window as it did on the fifth step (Figure 10).

7. Double click on the first error message. The editor window is opened, with focus placed on the problem line (line 2 of dhry_def.h). You may already fin out htat “#defin” should be replaced with “#define”. Instead of doing correctness in this way, we remove dhry_def.h from the project and supply it as a command line parameter

to the C compiler, which will be described in the next section.

1.1.3 Configuring the settings of build targets for your project

Build target settings must be selected separately for each build target in your project. To set build target options for the **dhryansi** example:

1. Ensure that the **DebugRel** build target is currently selected. By default, the DebugRel build target is selected when you create a new project based on the ARM project stationery. The currently selected build target is displayed in the **Build Target** pop-up menu in the project toolbar (Figure 10).

2. Select **Edit→DebugRel Settings… (or Alt + F7)**, as shown in Figure 12. The name of this menu item (Debug, Release, or DebugRel) changes depending on the name of the currently selected **Build Target**. The CodeWarrior IDE displays the **DebugRel Settings Panel** (Figure 14). All the target-specific settings are accessible through configuration panels listed at the left of the panel.
   An alternative way to do this step is to hit the *Build Target Setting* button in Project Window, as displayed in Figure 13.

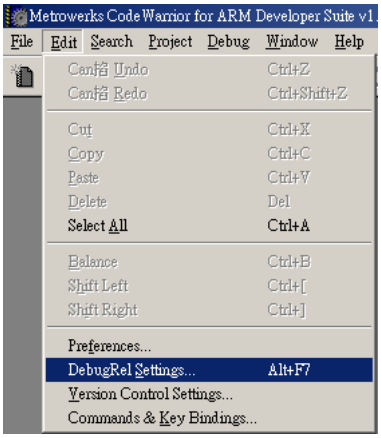

Figure 12. Select DebugRel Settings

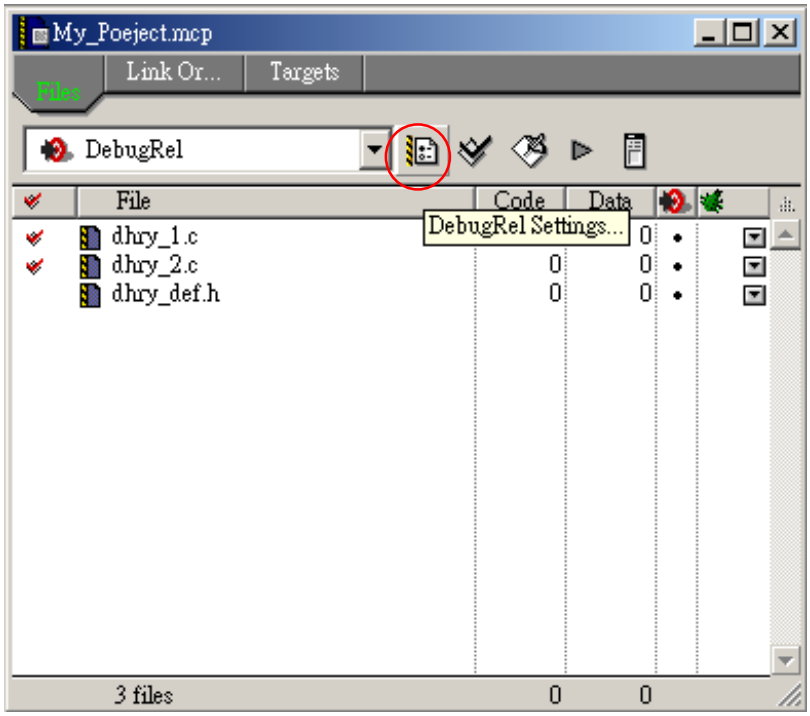

Figure 13. Select DebugRel Settings from Project Window.

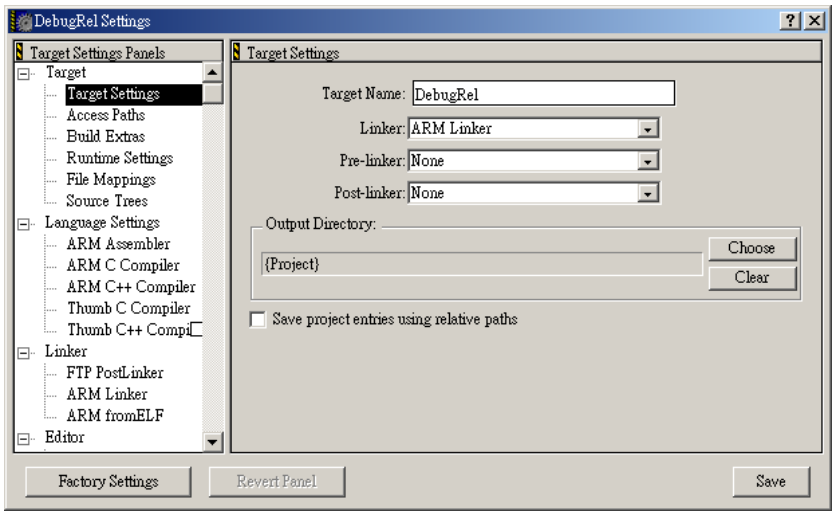

Figure 14. DebugRel Settings.

Click the *Access Paths* entry in the Settings Panels list. As displayed in Figure 13, the path {Compiler}Examples\dhryansi added in step 4 of Section 1.1.2 appears in the User Path. You can add other path by clicking the add button.

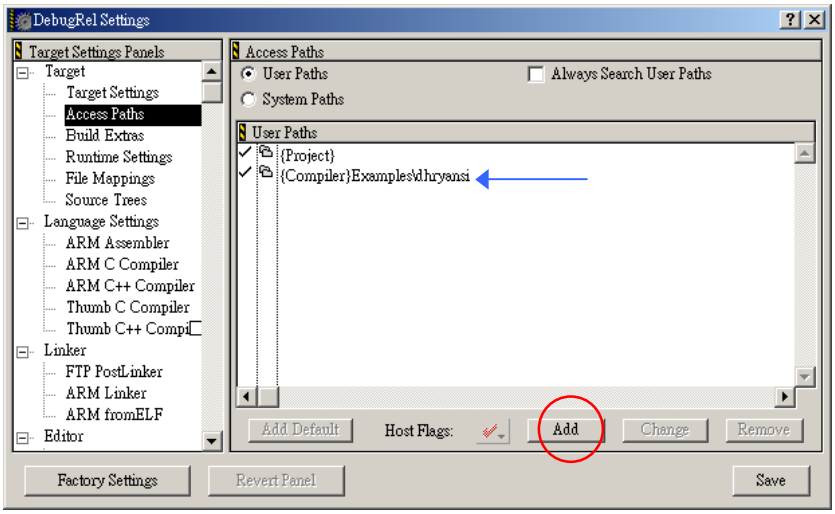

Figure 15. Access Path configuration

3. Click the **ARM C Compiler** entry in the Settings Panels list to display the configuration panel for the C compilers. The **Target and Source panel** is displayed. The panel consists of a number of tabbed panes containing groups of configuration options. For this example, the dhryansi source requires a *predefined macro* be set before it will compile.

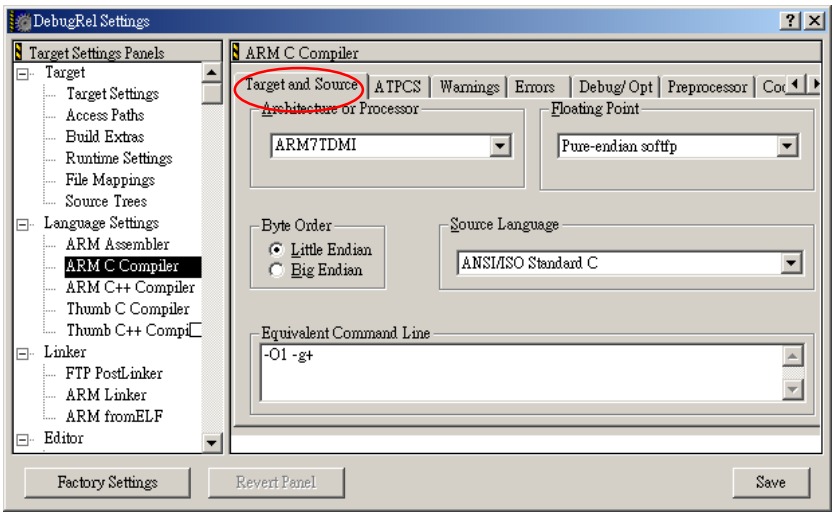

Figure 16. ARM C compiler panel.

4. Click the **Preprocessor tab** to display a list of predefined macros (Figure 17). Type **MSC_CLOCK** into the text field beneath the **List of #DEFINES** and **click Add** to define the MSC_CLOCK macro. In Figure 18, the CodeWarrior IDE adds MSC_CLOCK to the List of #DEFINES and the **Equivalent Command Line** text box displays the compiler command-line option required to define MSC_CLOCK.

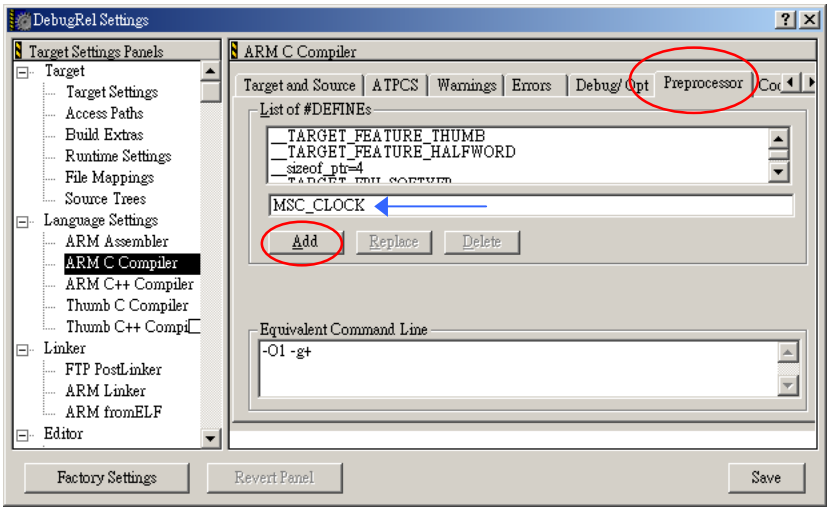

Figure 17. ARM C compiler preprocessor panel.

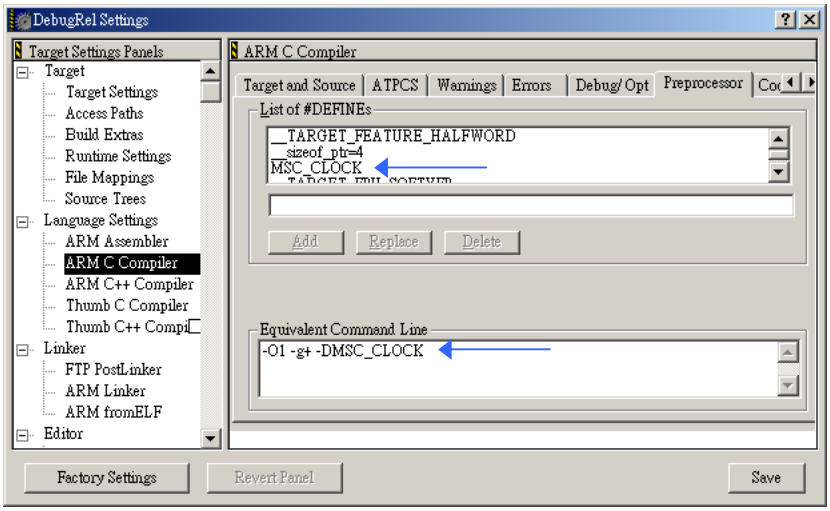

Figure 18. MSC_CLOCK defined.

6. **Click Save** to save your changes, and **close the DebugRel Settings panel**.

At this point you have defined the MSC_CLOCK macro for the DebugRel build target only. You must also define the MSC_CLOCK macro for the Release and Debug build targets if you want to use them. To select the **Release** build target:

1. Select **Release** from the **Build Target pop-up menu** (Figure 10) to change the current build target.
2. Apply the steps you followed above to define MSC_CLOCK for the Release build target.
3. Click on the **Debug/Opt tab** to display Debug and Optimization options for the Release build target. Select the **For time** Optimization Criterion button. The Equivalent Command Line text box reflects the change, as shown in Figure 19.

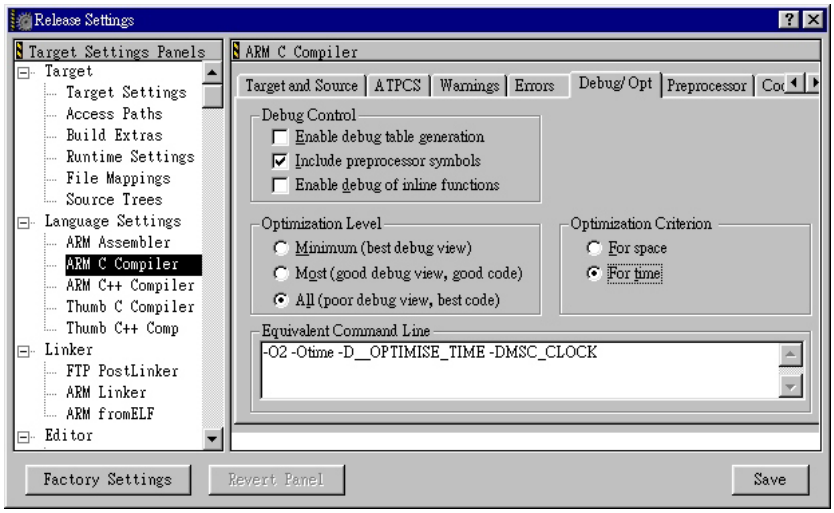

Figure 19. Debug/Opt configuration panel.

4. **Click Save** to save your settings.
5. Define MSC_CLOCK in the **Debug** build target in the same way as you have for the DebugRel and Release build targets. Your project is now equivalent to the dhryansi example project (PATH\ADSv1_1\Examples\dhryansi**dhryansi.mcp**) supplied with the ARM Developer Suite.

Note:

- Compiler options
  - -g Tells the compiler to add debug tables
  - -O1 Tells the compiler to select good optimization
  - -c Tells the compiler to compile only (not to link)
- There are configuration panels available for most of the ADS toolchain, including the linker, fromELF, and the assembler. You can use the configuration panels to specify most options available in the tools, including:
  - procedure call options
  - the structure of output images
  - the linker and postlinker to use
  - the ARM debugger to call from the CodeWarrior IDE.

  See the chapter on configuring a build target in the CodeWarrior IDE Guide for a complete description of build target options.

1.1.4 Building the project

The Project menu contains a number of commands to compile, or compile and link your project files. These commands apply only to the current build target. To compile and link the example project:

1. Select the build target you want to build (Figure 10). For this example, select the DebugRel build target.
2. Select **Project→Make** (or F7), as shown in Figure 20. The CodeWarrior IDE builds the project by:
   - compiling newly added, modified, and touched source files to produce ELF object files
   - linking object files and libraries to produce an ELF image file, or a partially linked object
   - performing any postlink operations that you have defined for your build target, such as calling fromELF to convert an ELF image file to another format. (Note:

In the dhryansi example there is no postlink operation.)

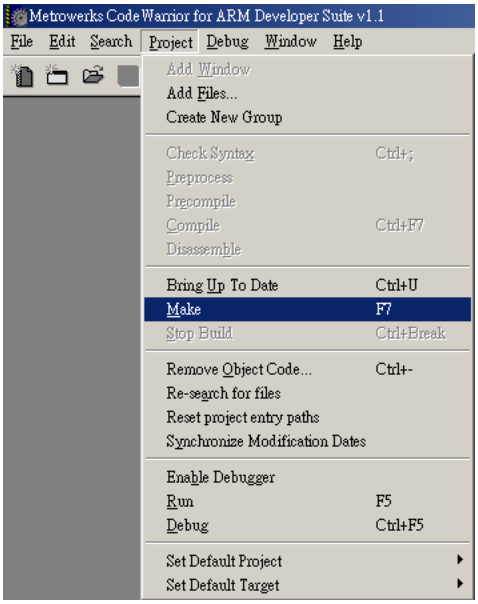

Figure 20. Make the project.

Note:

- An alternative way to Make the project is to click the Make button from the Project Window, as shown in Figure 21.
- If the project has already been compiled using a command such as **Bring Up To Date** or **Compile**, the Make command performs only the link and postlink steps.

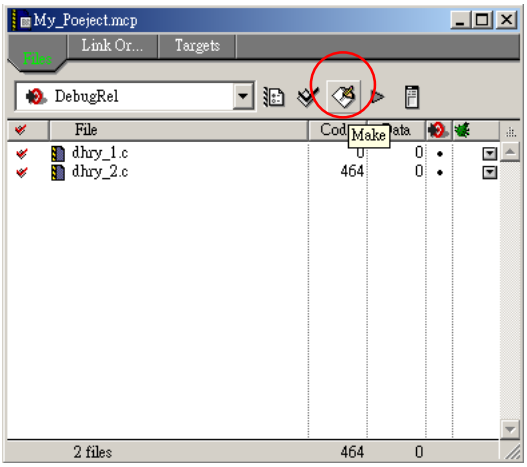

Figure 21. Make the project from the Project Window.

The compiler displays build information, errors, and warnings for the build in a messages window (Figure 22). The meaning of the “Image component sizes” will be explained later on.

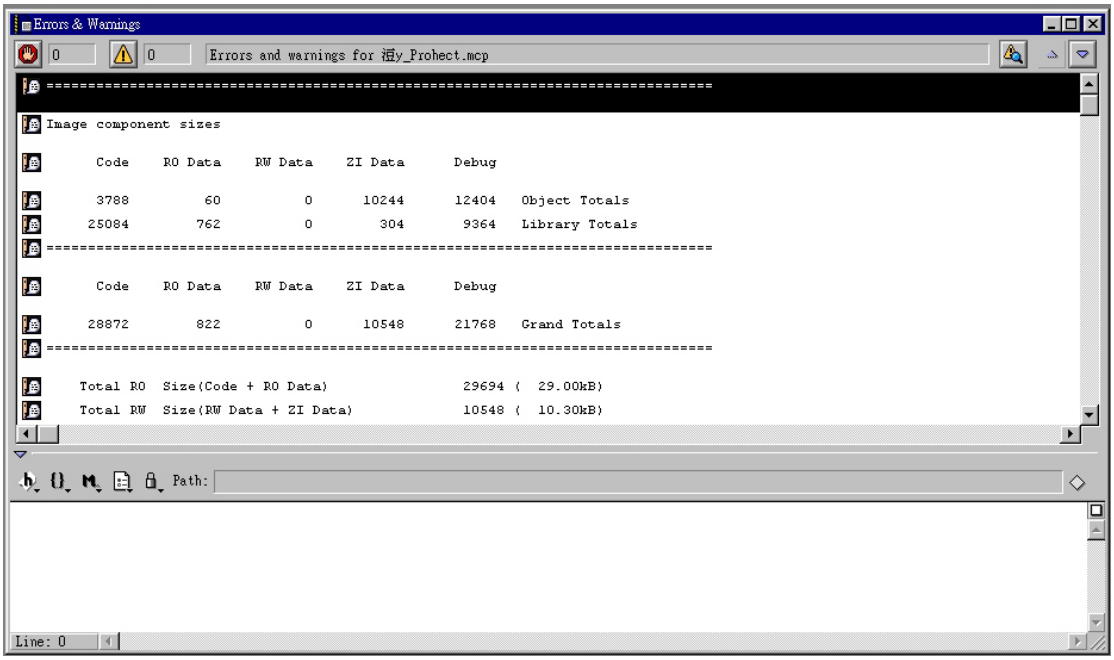

Figure 22. Errors & Warning message window.

After the last step, the following files are created.

PATH\My_project\My_project_Data\DebugRel\My_project.axf
PATH\My_project\My_project_Data\DebugRel\ObjectCode\dhry_1.o
PATH\My_project\My_project_Data\DebugRel\ObjectCode\dhry_2.o

3. Right-click on in the Project Window and select Disassemble from the pop-up menu. The disassembled code is displayed in a Disassembly window.

1.1.5 Debugging the project

To execute and debug your example project:

1. Select the build target you want to build (Figure 10). For this example, select the DebugRel build target.
2. Select **Project →Debug** (Ctrl + F5). The CodeWarrior IDE compiles and links any source files that are not up to date, and calls the **AXD** debugger to load the image and on standby to execute the image.

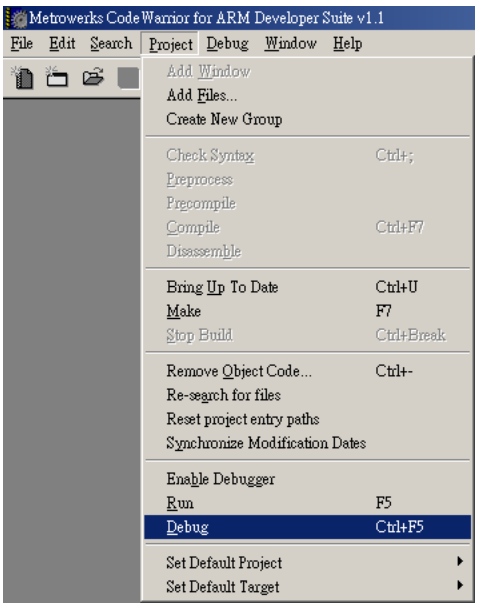

Figure 23. Select Debug.

Other ways to start AXD

- Click on **Run** button from Project Window, as shown in Figure 24. The CodeWarrior IDE then calls debugger to load and execute the image. The term ARM Runner refers to the ARM debugger that is called to execute, rather than debug, an image file.
- Double click on **My_project.axf**, AXD starts.
- Select **Start →Programs →ARM Developer Suite 1.1 →AXD Debugger**
- Using a Windows DOS shell: ***axd -debug filename.axf***

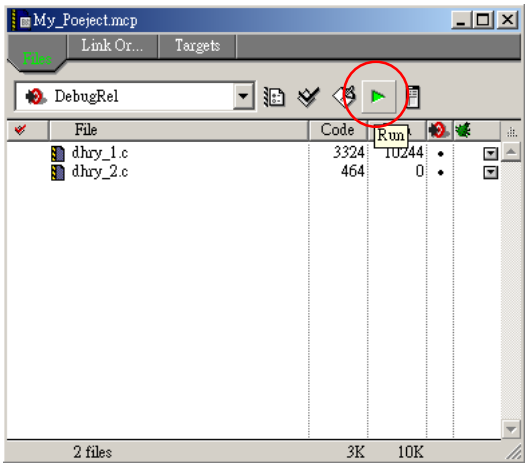

Figure 24. Debug the project from Project Window.

3. Select debugging system from **Options → Configure Target** (Figure 25). The AXD displays a **Choose Target Panel** (Figure 26). **Select ARMUL** and then **Click OK**.

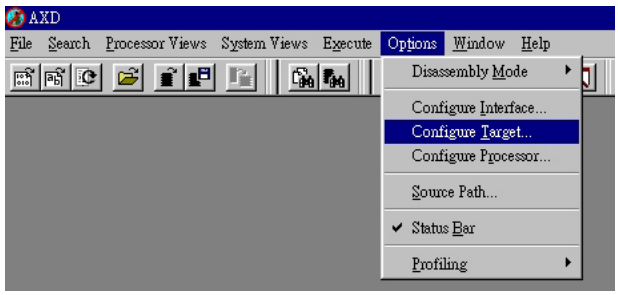

Figure 25. Configure Target.

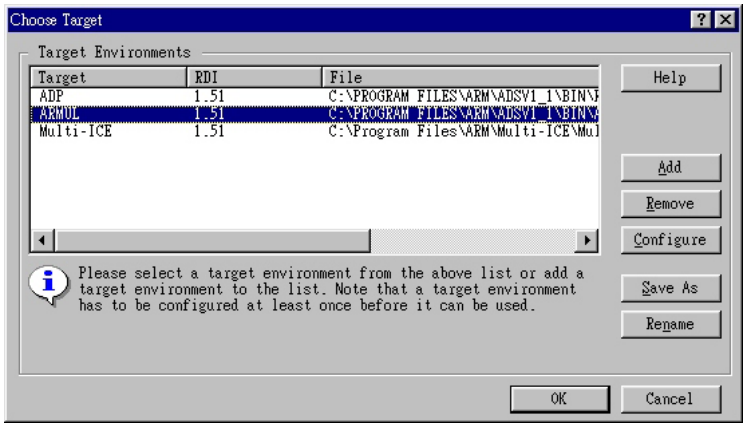

Figure 26. Choose Target Panel.

4. Use ***File →Load Images…*** to load a new image. If you start AXD from the CodeWarrior IDE, you can skip this step.

Figure 27. Loading an image.

5. Use ***Execute → Step*** (or F10) or the button shown in Figure 28 to step through the application. The disassembled code is displayed and a pointer indicates the current position (Figure 29). Use ***F10*** to execute the next instruction.

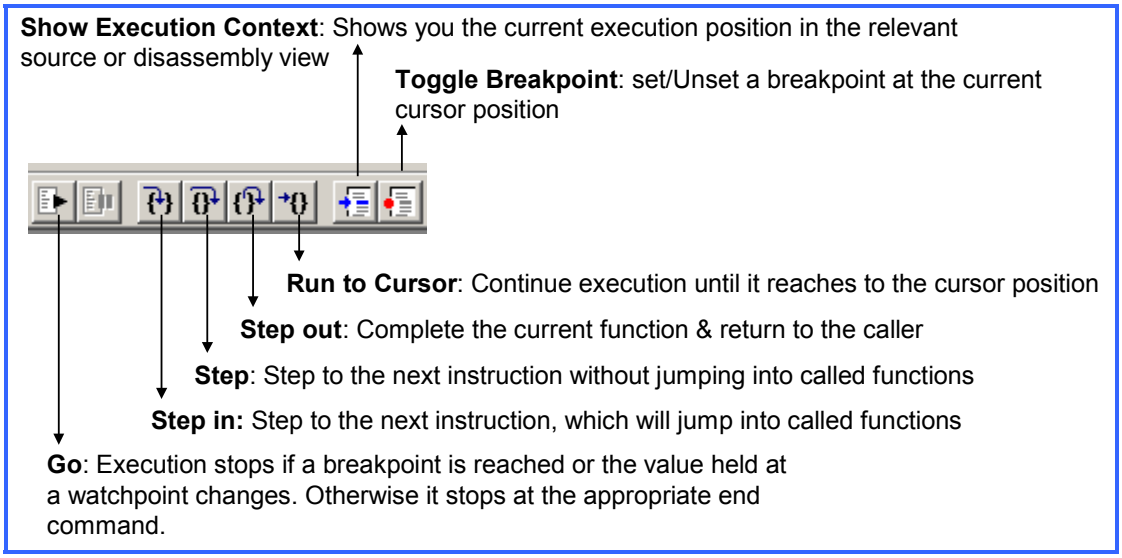

Figure 28. The Execute menu.

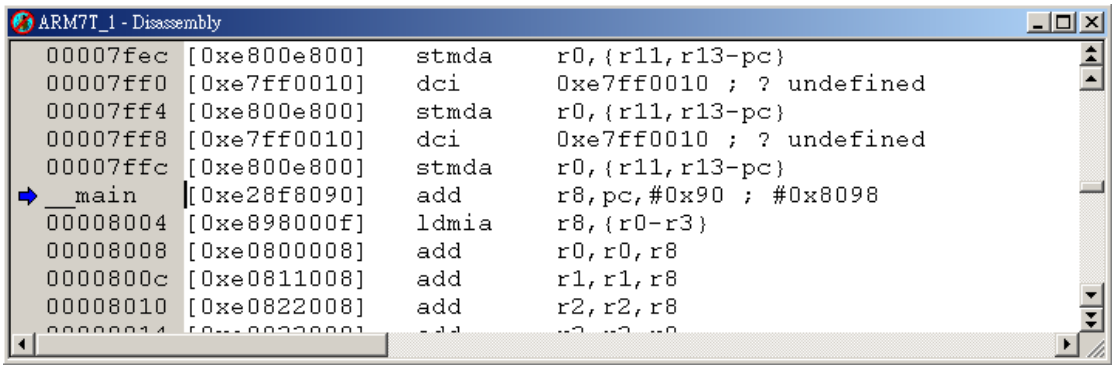

Figure 29. The disassembled code.

6. Interleaving source code.
   It is often useful to see interleaved code, i.e., the high level C code, and the low level assembled code together.
   6.1 Executing the image by ***selecting Execute ➔ Go*** or ***Go*** button in Figure 28. The program will run to the first breakpoint at main and the c/c++ source code will come into view.
   6.2 Right click on the c/c++ source code window and select ***Interleave disassembly***

Note:

- Before load an image, you have to make sure the selected target (ARMulator, Multi-ICE, EmbeddedICE or Angel debug monitor) exists, or you cannot load a image (*.axf).

Note: Explanation of File Extensions:

.c	C source file
.h	C header file
.o	object file
.s	ARM or Thumb assembly language source file
.mcp	ARM Project file, as used by the CodeWarrior IDE
.axf	ARM Executable file, as produced by armlink.
.txt	ASCII text file