In this tutorial, we’ll discuss how to use the IDE Debugger to examine and modify data. CodeWarrior enables you to manipulate your source code to display and change the contents of variables, arrays, and data structures. You can also use the Debugger to work at the hardware level if necessary.

Upon completion of this tutorial, you’ll be able to describe how to examine data variables and how to modify variable values. You’ll also be able to describe the low-level operations supported by the Debugger to view and modify sections of memory and the processor registers.
Data Basics

- Among other features, the Debugger can display --
  - all variables, both local and global.
  - the contents of memory.
  - the processor registers.

- Data appears in panes and separate windows.

- Variables have a name and a value.
  - Select the name or value to initiate actions.
  - Drag separating line to resize name/value columns.

- Values are almost always editable.

- Use hierarchy control for structures and pointers.

Let’s begin with a look at data and variables. You can use the Debugger to display a large amount of data. This data is organized into separate windows and then into panes for easy viewing and access. Typically, a glance at the Variables pane in the Thread Window will reveal the status of code execution.

Individual elements in an array or structure can be expanded by clicking the hierarchy control for the desired field. You can perform wide and deep disclosures of arrays and structured variables, depending on how the hierarchy controls are set.
Viewing Variables

- The primary location for viewing variables is the Thread Window’s Variables pane.
  - Pane opens when you launch code under Debugger control.
  - Pane typically shows variables for the function currently executing.

- When you select an item in the Stack Pane, its variables appear.
  - Displays current values
  - Pointer contents may have changed.

- The Variables pane displays global variables.
  - Displays the global variables that are used locally.
  - Display all with Window ==> Global Variables window command.

Variables are displayed in the Variables pane of the Main Debugger Thread Window. The variables pane shows all variables for the function currently executing. Clicking on a previous function in the Stack pane displays the variables for that function in the Variables pane.

There is also a very easy way to view Global variables, using the IDE’s Global Variables window. Click View > Global Variables to see these variables. Click on a source file to see variables that are global to that file.
Tracking a Variable: Expressions window

- To display a variable in the Expressions window,
  - select the name, then Data ==> Copy to Expression command. -- or--
  - right-click on the variable, then select Copy to Expression. -- or--
  - drag from any variable pane. -- or --
  - copy and paste the name to a new expression.

- Advantages over Variable window--
  - Information persists across debugging sessions.
  - Variables persist across scope boundaries.
  - There is no need to set-up a second time when re-entering a routine.
  - Multiple instances of the same variable are permitted.

Often it’s easier to track a variable if the variable is in its own window. To view a variable in its own window, you can double-click the variable name in the Variables pane of the Thread window, or right-click on the variable in the Source pane, and then select ‘View Variable’ from the submenu. Some disadvantages of this method are (1) you can only see one variable per window, (2) the variable disappears when it goes out of scope, and (3) the extra windows can clutter the screen.

Another way to track multiple variables at one time without creating a lot of screen “clutter” is to use the Expressions window. An advantage of the Expressions window is that it is persistent across multiple debug session. To open the Expressions window, click View > Expressions. Alternatively, click the Expressions button in the Thread window toolbar to open the Expressions window.

To add a variable to the expressions window, highlight the desired expression, then select Data > Copy to Expression, or drag and drop the highlighted expression into the Expressions window. If you are viewing a local variable in the Expressions window, its listing continues to exist through the life of the program. The displayed value is empty when the variable goes out of scope, but its current value is automatically restored when the variable comes back into scope.
Using the Expressions Window

• Use the View ==> Expressions window command to open.

• Use the Data ==> New Expressions command to enter a new expression.
  – Type the command in directly.
  – Copy or drag the command from code.
  – The expression Syntax follows C rules.

• Double-click an expression to edit.

• An expression can be --
  → a value (size of foo)
  → a variable (!my Variable)
  → a calculation (index + 12)

Expressions are one of the more powerful features of the Debugger. An expression represents a computation that produces a value. An expression can combine literal values, variables, registers, pointers, or a calculation.

To create an expression, open the Expressions window. Then, choose Data > New Expression to enter an expression. Double-click an expression to edit it.

Expressions use a specific syntax similar to the C programming language. You can perform typecasts of variables to see the result, compare strings, or calculate a value. However, you can’t call other functions from your code, use macros, assign values to variables, or increment/decrement variables.
Viewing Data Types

- Select the Data ➤ Show Types command sequence. --or-- Right-click the variable, then select Show Types from the pop-up menu.

- The data type will appear next to the name of the variable.

- You can customize your Debugger display preferences.
  - Select Edit ➤ Preferences ➤ Display Settings preference panel.
  - Optionally, show variable types by default.
  - Optionally, display dynamic types for objects.

Next, let’s take a closer look at variable data types. Every variable has an associated data type: short, double, char, and so forth. You can display a variable’s type in many different ways.

The easiest method is to use the Data > Show Types command, or Right-Click and select Show Types from the submenu. This command displays the data type for all variables, not just the selected variable.

To set the default display option, use the Display Settings panel to enable the Show variable types by default option.
Displaying a Value as a Different Type

• Select the variable value.
  – Select the Data Menu item for the desired data type.

• Right-click a variable, then select Data Type from the menu.

• Display multiple instances of the same variable --
  – in the multiple variable windows.
  – in the Expressions window.

• The Display Type command does not affect the actual data type.

Often while debugging, it’s useful to display a variable as a different type. For example, you might want to display the hexadecimal value of a decimal number. To do this, simply right-click the variable and choose the type of display you require.

Note that the display type does not affect the actual data type of the variable; it is merely displayed as a different type.
The CodeWarrior Debugger enables you to dynamically change the value of a variable while debugging. This can be useful in situations where you need to test a loop condition, for example.

To edit the variable, double-click the variable value in any Variable pane. Enter the new value in any accepted format, which can be decimal, hexadecimal, octal, binary, or character. The Debugger automatically converts the value to the correct type.
Viewing and Editing Data: Arrays

- Select Data ➞ View Array or right-click the variable required.
  - The display of all elements is automatic if the Debugger knows it; otherwise, the default size is 100 elements.
  - The user can set array size using the array window.

- For arrays of structures --
  - You can view the entire code structure or just one particular field of each element.

To view an array, select Data > View Array. The array variable will appear in its own window, including the size and a hierarchical list of its elements.

Use the Struct Member menu to select a particular field you need to view. This action enables you to filter out noise and see the same field in each element of an array or structure without being distracted by irrelevant data.
Viewing and Editing Data: Memory

- Select Data ➞ View Memory or right-click the variable.
  - Set the base address.
  - Edit the contents using hex or ASCII.
  - View memory as Raw Data, Disassembly, Source, or Mixed.

All variables reside at a specific memory address which is determined at run-time. If you want to view the memory address range of a variable, right-click the variable and choose View Memory from the shortcut menu.

You can also view a range of memory values as a specific type by using the Data > View Memory As command, then selecting the data type required.

You can view memory as Raw data, disassembly, source, or mixed, and you can directly edit the raw data using Hex or ASCII.
Viewing and Editing Data: Registers

- Use View ==> Registers commands to display registers.
  - Registers are grouped by function.
  - The registers shown very depending on the processors used.
- Directly edit a register by double-clicking on its value.

Register displays are platform-specific, since different chip architectures have different registers. You can see the registers for your platform by choosing View > Registers from the menu.
Viewing and Editing Register Details

• Right click on a specific Register in the Registers window.

• Select View ==> Register Details from the menu, and then specify the source XML file.
  – These files are located in bins, plug-ins, or support registers.
  – You can create your own description.

• View and enter information as you wish --
  – display in any format
  – see bit-fields and their meaning
  – read, write, and revert registers
  – reset a register to a defined value

Right-click on a register in the Registers window, and then select Register Details from the submenu, or use the View > Register Details command and then specify the desired register name, or browse for that register.

You can create your own register description files using XML format. Information on the required format is available from Metrowerks and will be provided in the standard documentation very soon.

Next, let’s look at the appearance of the Register Details window.
Viewing and Editing Register Details (cont’d)

The Register Details window is essentially an interactive help system for register information. You can read, write, and see all register information in a readable form.

This example shows the Data Address Register (DAR) of a PowerPC processor. By clicking on a bit in the display, you can obtain information on the bit’s purpose. Values below each bit position indicate whether the bit is set or not. Areas that are grey (if any) indicate reserved bits.

The pane beneath the register diagram can display different types of information. Here it provides detailed information on the purpose and function of the DAR register.
Let’s complete this tutorial with a couple of questions.

Which method do you use to make variables display persistently across debug sessions?

**Answer:**
The variables copied to the Expressions window persist across debug sessions.
Question

Which of the following low-level data operations can you perform using the Debugger? Click on your choice.

A) Edit the contents of memory using the View Memory window.
B) Edit the contents of a processor register using the Register window.
C) Edit the contents of a processor register using the Register Details window.
D) All of the above

Which of the following low-level data operations can you perform using the Debugger?

Answer:
You can manipulate any address in memory and all processor registers in the appropriate windows.
Tutorial Completion

• The CodeWarrior Debugger enables you to view and modify data in various ways.
  – at the source-code level using the names of variables, structures, or elements of structures
  – at low level, using memory addresses or processor registers

• You can display or enter data as different types.

• You can use the Expressions window to watch variables.
  – Variables displayed persist across debugging runs.
  – Local variables displayed persist even when the function goes out of scope.

Let’s review what we discussed in this tutorial.

The CodeWarrior Debugger provides many ways to view and manipulate data. You can examine data at the source code level, where you choose items of interest using variable names. This capability enables you to view complex structures and examine specific elements within a structure.

The CodeWarrior Debugger enables you to work at the hardware level if necessary. You can examine and edit a section of memory, or examine and modify the contents of a processor register.

You can use the Expressions window to obtain a persistent view of selected variables across debug sessions. You can track local variables that enter and leave scope as their function is entered or exited. This persistent display of crucial data, like CodeWarrior’s breakpoints, enables you to use multiple builds and debug sessions to quickly track down and fix problems with code.