CodeWarrior™ Development Studio

Embedded Systems
Development Tools
from Metrowerks

Metrowerks CodeWarrior™ Development Studio is a complete integrated development environment for hardware bring-up through programming embedded applications. By combining state-of-the-art debugging technology with the simplicity of a robust development environment, Metrowerks CodeWarrior Development Studio takes C/C++ source-level debugging and embedded application development to a new level.

The development studio provides a highly visual and automated framework that accelerates the development of even the most complex applications, so creating applications is fast and easy for developers of all experience levels.

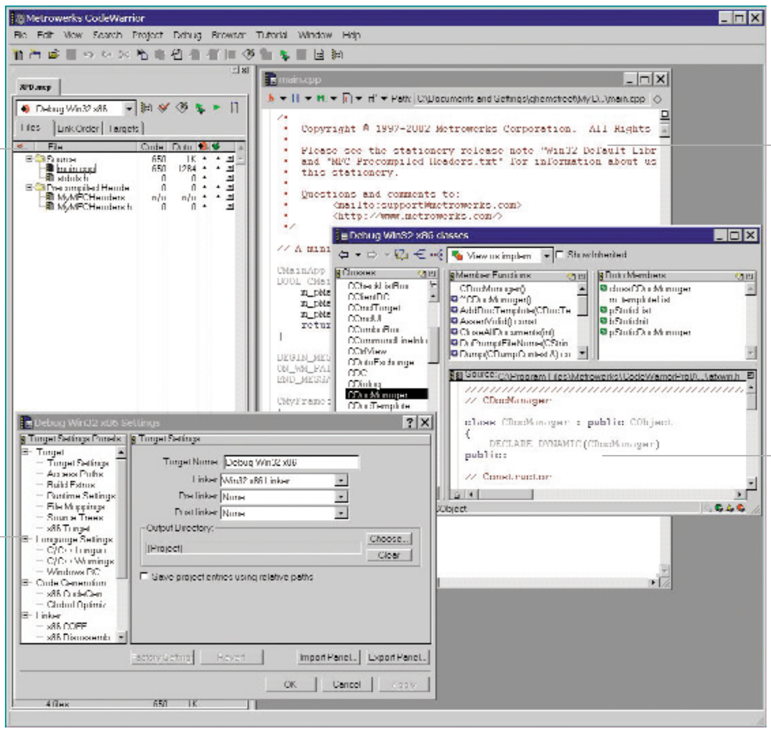
It is a single development environment that is consistent across all supported workstations and personal computers. On each of the supported platforms, the features and uses are identical. There is no need to worry about host-to-host incompatibilities.

The CodeWarrior Development Studio contains all of the tools needed to complete a major embedded development project:

- **Project Manager:** Handles top-level file management for the software developer; organizes project items by major group, such as files and targets; tracks state information (such as file modification dates); determines build order and inclusion of specific files in each build; coordinates with plug-ins to provide services like version-control and RTOS support.

*C++ not available for all architectures

- **Text Editor:** Enables the creation and manipulation of source code and other text files. Completely integrated with other IDE functions.
- **Search Engine:** Finds a specific text string replaces found text with substitute text; allows use of regular expressions; provides file-comparison and differencing functionality
- **Source Browser:** Maintains a symbolics database for the program; examples of symbols include names and values of variables and functions; uses the symbolics database to assist code navigation; links every symbol to other locations in the code related to that symbol; processes both object-oriented and procedural languages
- **Build System:** Uses the compiler to generate relocatable object code from source code and uses the linker to generate a final executable image from object code
  - **CodeWarrior C/C++* Compiler suite –** Includes the industry-leading C/C++* language CodeWarrior Compiler, including a Standard Template Library (STL) and a variety of other tools
- **Source-Level Debugger –** Provides a high-performance windowed source-level debugger equipped with the latest productivity enhancing graphical features to shorten board bring-up and application development time; Uses the symbolics database to provide source-level debugging; supports symbol formats such as CodeView, Debug With Arbitrary Records Format (DWARF), and STABS
- **Instruction Set Simulator –** Integrated instruction set simulator for jump-starting application development (Available for specific architectures only)



```
/*
 *  Copyright © 1997-2002 Metrowerks Corporation.  All Rights
 *
 *  Please see the stationery release note "Win32 Default Libr
 *  and "MFC Precompiled Headers.txt" for information about us
 *  this stationery.
 *
 *  Questions and comments to:
 *       <mailto:support@metrowerks.com>
 *       <http://www.metrowerks.com/>
 */
```

```
////////////////////////////////////////////
// CDocManager

class CDocManager : public CObject
{
     DECLARE_DYNAMIC(CDocManager)
public:

// Constructor
```

Target Name: Debug Win32 x86
Linker: Win32 x86 Linker
Pre-linker: None
Post-linker: None
Output Directory: {Project}

CodeWarrior Project Manager

The CodeWarrior Development Studio's Project Manager provides a powerful framework to simplify organizing, configuring, and building complex development projects; automating many aspects of managing a project.

The Project Manager component performs automatic dependency analysis and generates the appropriate project context. The powerful graphical user interface enables the user to configure a project by selecting from menus of options covering everything from optimization level, debugging level, and language-specific features to target type (executable or library) and much more. The Project Wizard takes the developer step-by-step through a series of questions to create a working project. Example stationery (a template) is provided as a starting place for the application. The stationery includes a linker command file and project files that makes it possible to associate debug connections easily. Stationery is provided for every supported CPU and programming language supported by the CodeWarrior Compiler.

CodeWarrior Text Editor

CodeWarrior Development Studio includes a full-featured, user-configurable, windowed text editor with features such as syntax coloring and auto-indenting. Syntax coloring helps quickly identify language keywords and constructs, including comments, strings, constants, and more. The CodeWarrior Text Editor implements all of the standard functions that are expected from an editor, including a powerful search feature that can find values within multiple files. The CodeWarrior Text Editor is fully configurable, so the developer can change the key bindings, font type, font size, color scheme, syntax coloring, and more. The CodeWarrior Text Editor also provides a single, consistent editor interface for all host and target development combinations. It's an integral part of the overall CodeWarrior Development Studio and can be invoked and controlled as an object from other components within the CodeWarrior Development Studio.

Search Engine

Industry observers estimate that software developers spend nearly half their time searching for basic information buried in application code. As applications grow in complexity, the time required to find, analyze, and modify code grows as a proportion of total engineering effort. The CodeWarrior Search Engine reduces this largely unproductive time by integrating code browsing and searching into a single tool.

The CodeWarrior Search Engine provides fast, semantic code navigation that makes it possible to find specific code structures, so finding a symbol or pattern among hundreds of directories and files is fast and easy.

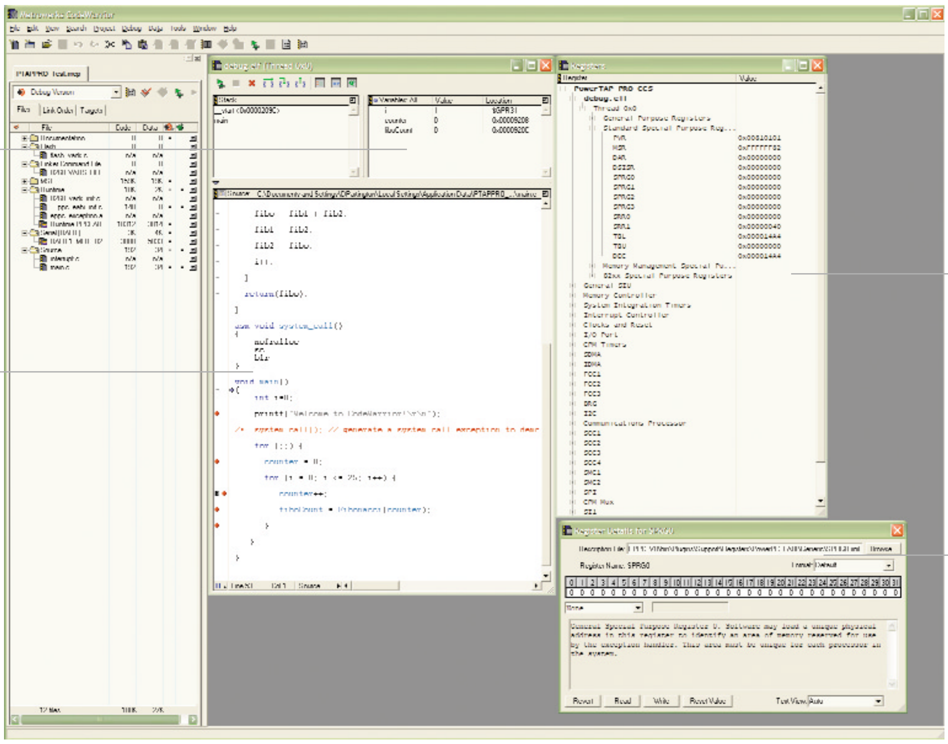
The seamless integration between the CodeWarrior Search Engine and the text editor means that all changes in the code are immediately reflected in the browser. No recompilation is necessary. With the CodeWarrior Search Engine, the mouse can be used to navigate between the different symbols. Just place the mouse cursor on a symbol and right-click to invoke the text editor, which will open the file and highlight the exact location of the selected symbol.

Plug-in Facility

The plug-in facility of the CodeWarrior Development Studio lets you extend it to include new features or to replace existing features. For example, you can develop a plug-in to create a new preference panel or you can write a plug-in that links the CodeWarrior Development Studio to a different compiler or linker.

We provide standard plug-in options for code management system interfaces like ClearCase and interfaces to other standard editors like Slik Edit, etc. Full documentation and source code is provided to assist you in creating your own plug-ins.

CodeWarrior Development Studio uses plug-ins to provide most of its services. For example, the standard compiler consists of a compiler plug-in with a small number of panel plug-ins to let users control its settings.



```
        fibo = fib1 + fib2;
        fib1 = fib2;
        fib2 = fibo;
        i++;
    }
    return(fibo);
}
asm void system_call()
{
        mfsrr0 r0
        sc
        blr
}
void main()
{
        int i=0;
        printf("Welcome to CodeWarrior!\r\n");
/*  system_call(); // generate a system call exception to demo
        for (;;) {
          counter = 0;
          for (i = 0; i <= 25; i++) {
              counter++;
              fiboCount = fibonacci(counter);
          }
        }
}
```

Variables: All	Value	Location
i	1	$GPR31
counter	0	0x00009208
fiboCount	0	0x0000920C

CodeWarrior Debugger

By combining a state-of-the-art IDE with the simplicity of a windowed environment, Metrowerks CodeWarrior Debugger takes C/C++ source-level debugging to a new level. The CodeWarrior Debugger assembles a wide array of high-powered components and features into a powerful graphical user interface to help get projects completed and to market ahead of schedule and under budget.

All of the CodeWarrior Debugger hardware and software features provide simple access and execution. Any debug operation desired is done through an intuitive "point-and-click" interface to make debugging fast, flexible, and easy.

Window-based Workspace Environment

The CodeWarrior Debugger enables developers to operate more efficiently with user friendly debugging, multiple windows, point-and-click capabilities and outline format.

CodeWarrior Debugger's interface allows users to customize the workspace to fit their needs: to create custom buttons, toolbars, and menus, and to "float" windows that are an integral part of the debugger so that they become independent windows on the workstation. This provides increased visibility and control over the display of information in the debugger. Windows that have been separated from the debugger can also be "docked" to rejoin the main debugger workspace controls.

The CodeWarrior Debugger's workspace allows users to focus on complex debugging tasks. Each workspace contains just the set of views needed for the task at hand. The application workspace provides a high-level view of the target software, while the hardware workspace provides a low-level view of the target hardware.

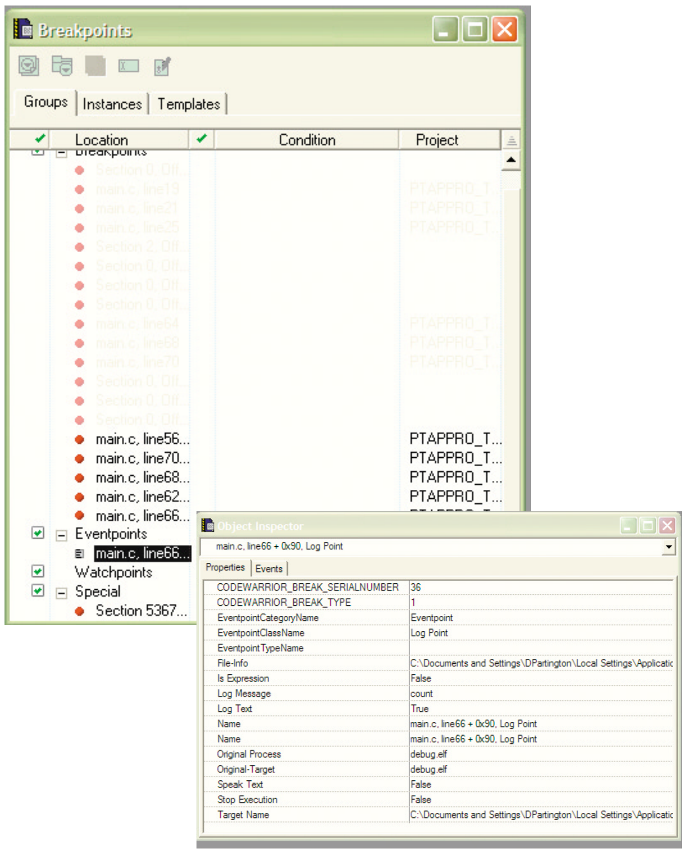
Seamless Integration

The CodeWarrior Debugger is fully integrated with a variety of run control devices like Metrowerks PowerTAP PRO and CodeTAP PRO, resulting in optimized run control and faster downloads.

Full-Featured Debugging

The CodeWarrior Debugger provides a rich set of debugging features designed to help the developer quickly find and repair software defects, including:

- **Breakpoints:** Breakpoints are easily set in source code by clicking on a "code-dot" in the window margin that indicates the point where breakpoint insertion is possible. Once the breakpoint has been set, the "code-dot" changes appearance to indicate the setting. Removal of the breakpoint is just as easy; simply click the "dot" and the breakpoint is automatically removed. Performing a right click on the breakpoint enables the behavior of the breakpoint to be changed to make it conditional. It can also be changed to a hardware breakpoint, or attached as an action to the breakpoint that is performed once the breakpoint is hit.
- **Eventpoints:** Eventpoints are used to perform a task when program execution arrives at a specific line of source code or when an associated conditional expression evaluates to true. You can set an eventpoint that performs a task such as running a script, playing a sound, or collecting trace data. An eventpoint is equivalent to a breakpoint that performs a task other than halting program execution. Eventpoints are:
  - **Log Point -** Logs or speaks a string or expression and records messages to the Log window
  - **Pause Point -** Pauses execution just long enough to refresh debugger data
  - **Script Point -** Runs a script, application, or other item



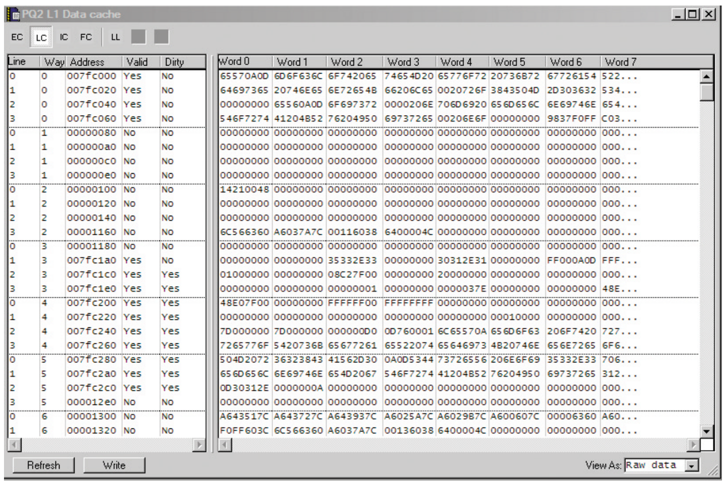
✓	Location	✓	Condition	Project
✓	Breakpoints			
	● Section 0, Off...			
	● main.c, line19			PTAPPRO_T...
	● main.c, line21			PTAPPRO_T...
	● main.c, line25			PTAPPRO_T...
	● Section 2, Off...			
	● Section 0, Off...			
	● Section 0, Off...			
	● Section 0, Off...			
	● main.c, line64			PTAPPRO_T...
	● main.c, line68			PTAPPRO_T...
	● main.c, line70			PTAPPRO_T...
	● Section 0, Off...			
	● Section 0, Off...			
	● Section 0, Off...			
	● main.c, line56...			PTAPPRO_T...
	● main.c, line70...			PTAPPRO_T...
	● main.c, line68...			PTAPPRO_T...
	● main.c, line62...			PTAPPRO_T...
	● main.c, line66...			
✓	Eventpoints			
	main.c, line66...			
✓	Watchpoints			
✓	Special			
	● Section 5367...			

main.c, line66 + 0x90, Log Point

CODEWARRIOR_BREAK_SERIALNUMBER	36
CODEWARRIOR_BREAK_TYPE	1
EventpointCategoryName	Eventpoint
EventpointClassName	Log Point
EventpointTypeName	
File-Info	C:\Documents and Settings\DPartington\Local Settings\Applicatic
Is Expression	False
Log Message	count
Log Text	True
Name	main.c, line66 + 0x90, Log Point
Name	main.c, line66 + 0x90, Log Point
Original Process	debug.elf
Original-Target	debug.elf
Speak Text	False
Stop Execution	False
Target Name	C:\Documents and Settings\DPartington\Local Settings\Applicatic

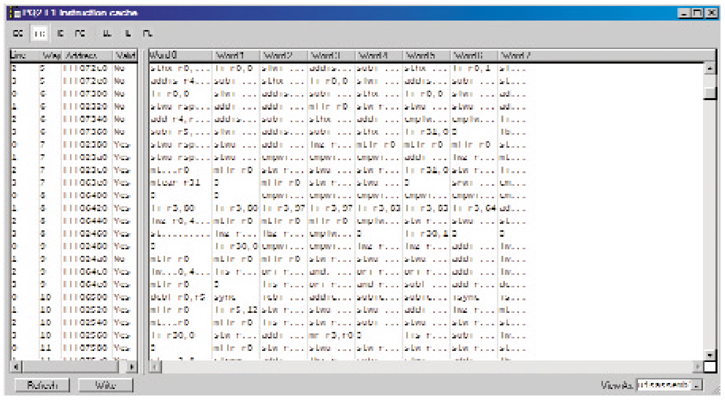
  - **Skip Point -** Skips execution of a line of source code
  - **Sound Point -** Plays a sound
  - **Trace Collection Off -** Stops collecting trace data for the Trace Window
  - **Trace Collection On -** Starts collecting trace data for the Trace Window
- **Watchpoints:** Watchpoints halt program execution when a specific location in memory changes value. After you set a watchpoint at a key point in memory, you can halt program execution when that point in memory changes value or, for some devices, when the memory location is accessed, examine the call chain, check register and variable values, and step through your code. You can also change values and alter the flow of normal program execution. A watchpoint is equivalent to a memory breakpoint. Watchpoints states:
  - **Enabled -** Indicates that the watchpoint is currently enabled. The debugger halts program execution at an enabled watchpoint.
  - **Disabled -** Indicates the watchpoint is currently disabled. The debugger does not halt program execution at a disabled watchpoint. Use the Condition column of the Breakpoints window to set a conditional watchpoint. A conditional watchpoint has an associated conditional expression. The debugger evaluates the expression to determine whether to halt program.
- **Special Breakpoints**: Special breakpoints halt program execution for very specific reasons:
  - Program execution arrives at the beginning of the function main()
  - A C++ or Java exception occurs
  - An event occurs that the debugger plug-in defines as a break event You cannot change or delete special breakpoints, but you can enable and disable them.

- **Single-stepping:** The CodeWarrior Debugger supports the following single-stepping mechanisms:
  - **Step Into –** Traces execution of every individual instruction
  - **Step Over –** Does not trace into the called function
  - **Step Out –** Brings execution back to the calling function
- **Tooltips –** Enables the developer to view crucial information easily. Data Tooltips display a quick, one-time view of a variable, while Icon Tooltips display an item's function when the cursor is placed over it.
- **Variable View on Mouse Over:** Get the current value of a specific variable in the source display
- **Simple module and function browsing:** Enables access to an internal table of all modules, global variables, and functions in a given debug context. With a single right-click, it's possible to edit code, run it to a target address or set breakpoints at entry or exit.
- **Display stack trace:** Provides an easy display of all procedures (functions) active in the calling chain, and enables the developer to follow the progress of a program through its hierarchical call structure. The trace information includes the name of each procedure, module name, line number at source-level, physical address in memory and the name and value of each argument.
- **Local variables display:** Shows the variables local to the current function. As running code moves from function to function, the contents of the local variables view change to display the local variables of the current function being viewed.
- **Displaying data:** Offers three ways to view data:
  - **Data Tooltip –** Displays the values of a variable directly from source code.
  - **Instant Watch –** Provides a view of the variable's data in a popup view that allows pointers to be followed.
  - **Watch view –** Allows for monitoring and updating data in a separate window.
- **Memory view:** Gives programmers the ability to display and modify the contents of target memory. Features include automatic alignment, find in memory, the ability to compare two memory regions or memory and a file, uploading memory contents to a file, filling memory with a known value, freezing the memory view to prevent target access, invoking multiple memory views, cutting, pasting and more. Memory can be formatted in a variety of ways, including hexadecimal, decimal, octal, ASCII, binary, big/little endian among others.
- **Register view:** Provides extensive information on CPU core and peripheral registers, as well as up to 128 user-defined custom registers. All registers displayed can also include bit level details on register contents. Bit level details are formatting details that break down and describe the contents of bit-mapped registers, making it easy to interpret the register contents. Bit field values are displayed as English-language equivalents of bit field patterns. Picking a value from a pull-down list or manually entering the register value can result in bit field value changes within the registers.
- **Cache view:** View cache information for the target processor.
- **Object file format:** Supports STABS and, ELF/DWARF 1 and 2 object file output formats.
- **Multi-core/CPU debugging:** Enables debugging of multi-core System on Chip (SoC) and multiple-CPU targets. Every core has its own independent register view, memory view, stack view, disassembly view, source view and more. It's not necessary to run multiple instances of the debugger (as other products require). And it's possible to debug a mixture of different types of cores. Features like "stop all cores" and "run all cores," as well as single-stepping some cores while the other cores are running provide the power to synchronize and control debugging sessions



Line	Way	Address	Valid	Dirty	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
0	0	007fc000	Yes	No	65570A0D	6D6F636C	6F742065	74654D20	65776F72	20736B72	67726154	522...
1	0	007fc020	Yes	No	64697365	20746E65	6E72654B	66206C65	0020726F	3843504D	2D303632	534...
2	0	007fc040	Yes	No	00000000	65560A0D	6F697372	0000206E	706D6920	656D656C	6E69746E	654...
3	0	007fc060	Yes	No	546F7274	41204B52	76204950	69737265	00206E6F	00000000	9837F0FF	C03...
0	1	00000080	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
1	1	000000a0	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
2	1	000000c0	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
3	1	000000e0	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
0	2	00000100	No	No	14210048	00000000	00000000	00000000	00000000	00000000	00000000	000...
1	2	00000120	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
2	2	00000140	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
3	2	00001160	No	No	6C566360	A6037A7C	00116038	6400004C	00000000	00000000	00000000	000...
0	3	00001180	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
1	3	007fc1a0	Yes	No	00000000	00000000	35332E33	00000000	30312E31	00000000	FF000A0D	FFF...
2	3	007fc1c0	Yes	Yes	01000000	00000000	08C27F00	00000000	20000000	00000000	00000000	000...
3	3	007fc1e0	Yes	Yes	00000000	00000000	00000001	00000000	0000037E	00000000	00000000	48E...
0	4	007fc200	Yes	Yes	48E07F00	00000000	FFFFFF00	FFFFFFFF	00000000	00000000	00000000	000...
1	4	007fc220	Yes	Yes	00000000	00000000	00000000	00000000	00000000	00010000	00000000	000...
2	4	007fc240	Yes	Yes	7D000000	7D000000	000000D0	0D760001	6C65570A	656D6F63	206F7420	727...
3	4	007fc260	Yes	Yes	7265776F	5420736B	65677261	65522074	65646973	4B20746E	656E7265	6F6...
0	5	007fc280	Yes	Yes	504D2072	36323843	41562D30	0A0D5344	73726556	206E6F69	35332E33	706...
1	5	007fc2a0	Yes	Yes	656D656C	6E69746E	654D2067	546F7274	41204B52	76204950	69737265	312...
2	5	007fc2c0	Yes	Yes	0D30312E	0000000A	00000000	00000000	00000000	00000000	00000000	000...
3	5	000012e0	No	No	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000...
0	6	00001300	No	No	A643517C	A643727C	A643937C	A6025A7C	A6029B7C	A600607C	00006360	A60...
1	6	00001320	No	No	F0FF603C	6C566360	A6037A7C	00136038	6400004C	00000000	00000000	000...

View As: Raw data



to match the target behavior. The CodeWarrior Debugger is designed to support multiple debug sessions running on one or more hosts simultaneously.

- **Mixed language debugging:** Supports mixed language debugging in C, C++, and Assembly Language. When moving between source modules written in different languages, the CodeWarrior Debugger automatically analyzes the language of the file in view and adjusts the expression evaluation and data display accordingly.
- **Target connection wizard:** Simplifies and automates the task of defining new connection definitions based on hardware and communication parameters.
- **Profile Window:** Examine profile data that you collect from executing code. Examining this data helps you improve the performance of your project. You use profiler API or #pragma directives in your source code to turn on the profiler, collect profiling data and turn off the profiler.*
- **Command-Line Window:** Supports a command-line interface to some of its features. You can use the command-line interface together with various scripting engines, such as the Microsoft® Visual Basic® script engine, the Java™ script engine, TCL, Python, and Perl. You can also issue a command line that saves a log file of command-line activity.

Board bring-up

The CodeWarrior Debugger helps developers deal with the complexity of bringing up a board by providing complete control over all board settings, including initial register values and memory configuration. After initial target register values are defined, the debugger restores these values each time the user connects to their board. Then an assembler source file can be created from these settings as an addition to the project. The CodeWarrior Debugger also includes a comprehensive set of hardware diagnostics and robust flash programming to support an extensive list of flash devices.

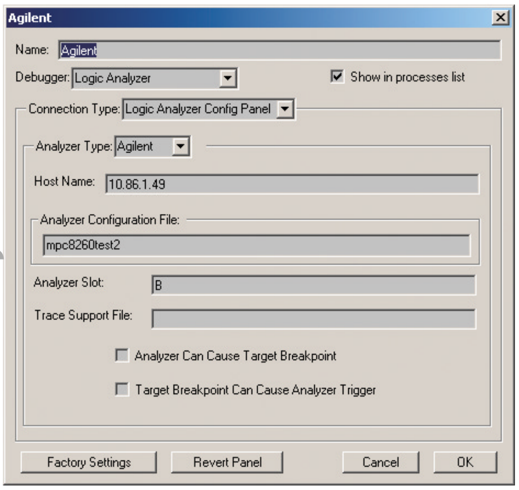
* Not available on all Architectures.

Flash programming

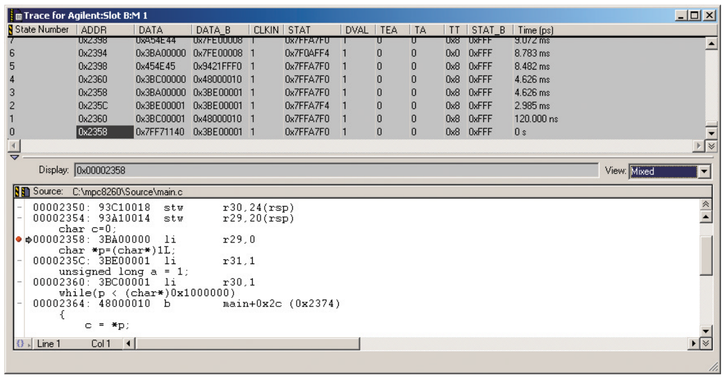
Program on-board Flash devices from within the same graphical user interface used to troubleshoot the application. No boot code is required to run on the target system in order to use the programming features of the CodeWarrior Flash Programmer.

Logic Analyzer

At the most complex of hardware development is the need to troubleshoot low-level hardware components. This type of activity gives rise to the need for developers to utilize the CodeWarrior Debugger in concert with a Logic Analyzer to understand complex signals on an embedded hardware platform.


Agilent

Name: Agilent
Debugger: Logic Analyzer
[✓] Show in processes list
Connection Type: Logic Analyzer Config Panel
Analyzer Type: Agilent
Host Name: 10.86.1.49
Analyzer Configuration File: mpc8260test2
Analyzer Slot: B
Trace Support File:
[ ] Analyzer Can Cause Target Breakpoint
[ ] Target Breakpoint Can Cause Analyzer Trigger


Trace for Agilent:Slot B:M 1

State Number	ADDR	DATA	DATA_B	CLKIN	STAT	DVAL	TEA	TA	TT	STAT_B	Time (ps)
7	0x2398	0xA454E44	0x7FE00008	1	0x7FFA7F0	1	0	0	0x8	0xFFF	9.072 ms
6	0x2394	0x3BA00000	0x7FE00008	1	0x7F0AFF4	1	0	0	0x0	0xFFF	8.783 ms
5	0x2398	0x454E45	0x9421FFF0	1	0x7FFA7F0	1	0	0	0x8	0xFFF	8.482 ms
4	0x2360	0x3BC00000	0x48000010	1	0x7FFA7F0	1	0	0	0x8	0xFFF	4.626 ms
3	0x2358	0x3BA00000	0x3BE00001	1	0x7FFA7F0	1	0	0	0x8	0xFFF	4.626 ms
2	0x235C	0x3BE00001	0x3BE00001	1	0x7FFA7F4	1	0	0	0x8	0xFFF	2.985 ms
1	0x2360	0x3BC00001	0x48000010	1	0x7FFA7F0	1	0	0	0x8	0xFFF	120.000 ns
0	0x2358	0x7FF71140	0x3BE00001	1	0x7FFA7F0	1	0	0	0x8	0xFFF	0 s

Display: 0x00002358 View: Mixed

Source: C:\mpc8260\Source\main.c

```
- 00002350: 93C10018  stw      r30,24(rsp)
- 00002354: 93A10014  stw      r29,20(rsp)
     char c=0;
● ⇨00002358: 3BA00000  li       r29,0
     char *p=(char*)1L;
- 0000235C: 3BE00001  li       r31,1
     unsigned long a = 1;
- 00002360: 3BC00001  li       r30,1
     while(p < (char*)0x1000000)
- 00002364: 48000010  b        main+0x2c (0x2374)
     {
          c = *p;
```

Line 1 Col 1

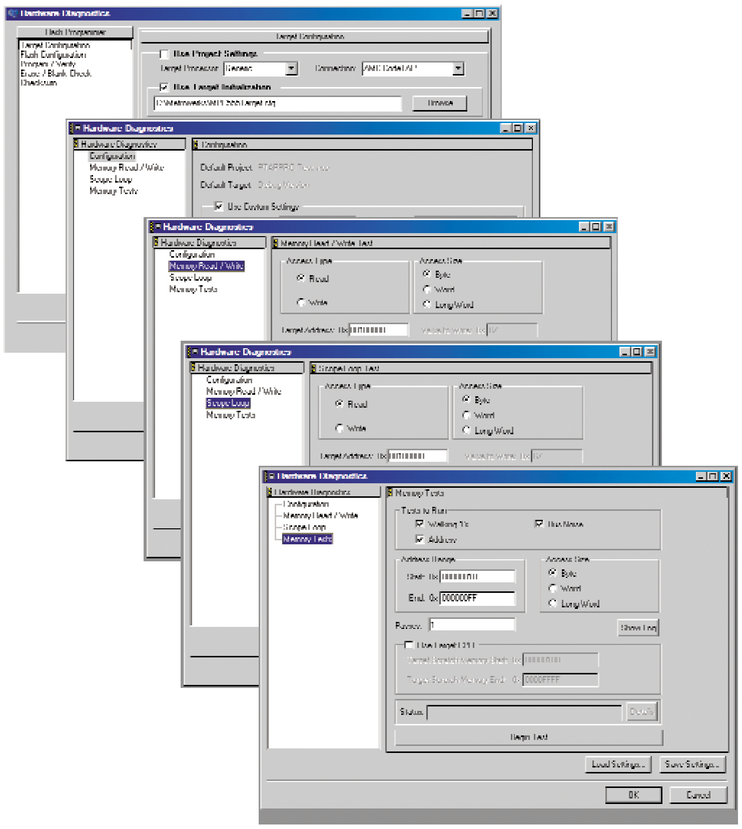
Metrowerks has implemented such an interface to seamlessly integrate Logic Analyzer communications into the CodeWarrior Debugger. Features included are:

- Trace On/Off
- Trace Everything
- Trace History
- Start Trace Based on Specified Address
- Start Trace on Address Range
- Trace All in Address Range
- Breakpoint on Trigger
- Trigger Tracing on Breakpoint
- Support for: Tektronix and Agilent

Hardware Diagnostics

The CodeWarrior Development Studio comes with diagnostics that enable the developer to determine if the basic hardware is functional. These tests include:

- Memory Read / Write: The Memory Read / Write component performs diagnostic tests for performing memory reads and writes over the remote connection interface
- Scope Loop: The Scope Loop component configures diagnostic tests for performing repeated memory reads and writes over the remote connection interface. The tests repeat until you stop them. By performing repeated read and write operations, you can use a scope analyzer or logic analyzer to debug the hardware device
- Memory Tests: The Memory Tests component lets you perform three different tests on the hardware:
  - Walking Ones
  - Address
  - Bus Noise



You can specify any combination of the tests and the number of passes to perform them. For each pass, the hardware diagnostic tools perform the tests in turn, until all passes are complete. The tools tally memory test failures and display them in a log window after all passes are complete. Errors resulting from memory test failures do not stop the testing process; however, fatal errors immediately stop the testing process.

CodeWarrior Compiler

The Metrowerks CodeWarrior Development Studio combines industry leading components to offer the embedded developer all the necessary tools to create, build, and deploy quality products to their customers. One major component of the IDE is the CodeWarrior Compiler. It combines industry-proven optimization technology with the versatility and control needed to fully exploit today's complex PC CPUs. The CodeWarrior Compiler's design is based on a partitioned architecture that results in proven reliability and flexibility for embedded applications, as well as interoperability with other CodeWarrior development products.

The CodeWarrior Compiler provides language-specific front ends for C and C++ that parse the original source code into a common token-based representation of the source. Optimizations are applied to this intermediate language representation. Also, the fully optimized code is converted into the appropriate machine code via a robust, table-driven back-end module. Metrowerks' close relationship with silicon partners, combined with the CodeWarrior Compiler's modular design, make it possible for the CodeWarrior portfolio to provide highly optimized compilers for new silicon with very short lead times. The CodeWarrior compiler's modular architecture enables you to immediately gain maximum performance from your compiler/silicon investment.

Proven Optimization Technology

Metrowerks CodeWarrior compiler produces exceptionally fast, compact, high-quality object code. A large number of highly refined, global, local, CPU-specific, and application–specific (profile-driven) optimization techniques enable the programmer to fine-tune the compiler's output to match the application's requirements. Programmers can select various optimizations to balance execution speed with code size while intelligent defaults can generate optimal code out of the box.

Advanced C/C++ Compiler – Designed for highly embedded development support. Key features include:

- Advanced optimization technology generates fast, compact, high-quality code
- Field-proven reliability to meet extreme embedded design constraints
- Compatibility with the latest ANSI C++ specs (ISO/IEC 14882:1998E) and the ANSI C spec (X3.159-1989)
- Standards conformance (ANSI and EABI) for maximum tool interoperability
- Complete control of code and data memory allocation
- Options to pack or byte-swap structures to match existing data types
- Supports position independent code (PIC) and data (PID)
- Board support routines for bare board applications (no OS)
- Proven performance with industry leading RTOSes

Assembler – full-featured macro assembler that is invoked automatically by the Project Manager or as a complete standalone assembler for generating object modules. Key features include:

- Conditional macro assembler with over thirty directives
- Unlimited number of symbols
- Debug information for source level debugging of assembly programs

Linker – offers precise control over the allocation, placement, and alignment of code and data in memory. Key features include:

- Links object modules into absolute or relocatable modules
- Reads/ writes/ mixes ELF and STABS object files
- Generates fully EABI compliant ELF/ DWARF 2.0 output for consumer tool interoperability

Libraries – the Metrowerks Standard Libraries is included:

- Complete C++ library (STL)
- Complete, reentrant C libraries compliant with ANSI/ISO, POSIX, and SVID standards
- Multithreading
- Full complement of math libraries, including IEEE-754 Appendix functions
- Efficient floating-point libraries for fast execution of calculations

Profiler – profiling options contained in the compiler instrument application code, which when executed save profile information that can be viewed by the profiler utility. This profile data can also be automatically to the compiler for additional code optimization based on execution paths.

Documentation – the IDE and CodeWarrior compiler ship with extensive documentation specific to your chosen architecture. Getting Started Guide enables you to quickly get up-to-speed and enhances out-of-box experience. In addition to hardcopy, all manuals are available in HTML and PDF formats.

CodeWarrior Instruction Set Simulator

The CodeWarrior Instruction Set Simulator provides a quick and easy way to begin developing code without the requirement for access to hardware. The ability to develop software without requiring hardware provides a number of significant benefits to software engineers, including the ability to run code before custom hardware is available, running/testing code when hardware resources are limited, and learning how to use the development environment without first having to get hardware running.

The CodeWarrior ISS provides full instruction simulation and fully supports standard C library I/O. It is fully integrated with the CodeWarrior IDE, and also provides a full command-line interface. The CodeWarrior ISS is available for specific platforms.

DS91516A

Metrowerks United States	Metrowerks Europe	Metrowerks Japan
7700 West Parmer Lane	Metrowerks GmbH	Shibuya Mitsuba Bldg. 5F Udagawa-cho 20-11
Austin, TX 78729	Schatzbogen 7, D-81829 München	Shibuya-ku Tokyo 150-0042 Japan
Phone +1.512.996.5300	Phone: +49 611 3611 850	Phone +81.3.3780.6091
Fax: +1.512.996.4910	Fax: +49 611 3611 85 1	Fax +81.3.3780.6092
E-mail: info@metrowerks.com	E-mail: info_europe@Metrowerks.com	E-mail: asia-sales@metrowerks.com

metrowerks