RealMonitor™ Support and Program Caching in SourcePoint™

Overview

The most common technique for debugging embedded applications is to control the processor with a debugger, using either a hardware control device or a debug agent running on the device under test. Debuggers allow the user to run, stop, step and set breakpoints.

This method of debugging is effective, but is not always feasible. Some embedded systems, such as control systems, can damage equipment or even cause undesirable conditions if they are stopped by a debugger. Embedded systems developers must employ other methods to debug these types of applications.

Users of ARM processors have at their disposal two debugging modes that allow the user to debug applications while the target continues to run and service interrupts: 1) RealMonitor™ and 2) Program caching.

RealMonitor, an ARM Ltd. product that you embed into your target code, allows writing and reading data to/from the target while the processor continues to run. The American Arium SourcePoint™ debugger, connected to an Arium emulator, supports this mode of operation. This allows the embedded systems developer to debug applications "on the fly."

In addition, American Arium emulators can provide visibility to processor execution while the processor is running via program caching. By keeping a cached copy of the program running on the target, SourcePoint is able to disassemble ETM trace while the target is running. This functionality is independent of RealMonitor.

RealMonitor Support

Purpose

Designers of products that contain complex control systems often require debugging tools that provide debugging functionality without stopping the target system. As these control systems are usually interrupt driven, RealMonitor is an ideal solution. Users can fully debug their systems without creating undesirable conditions caused by stopping the control system.

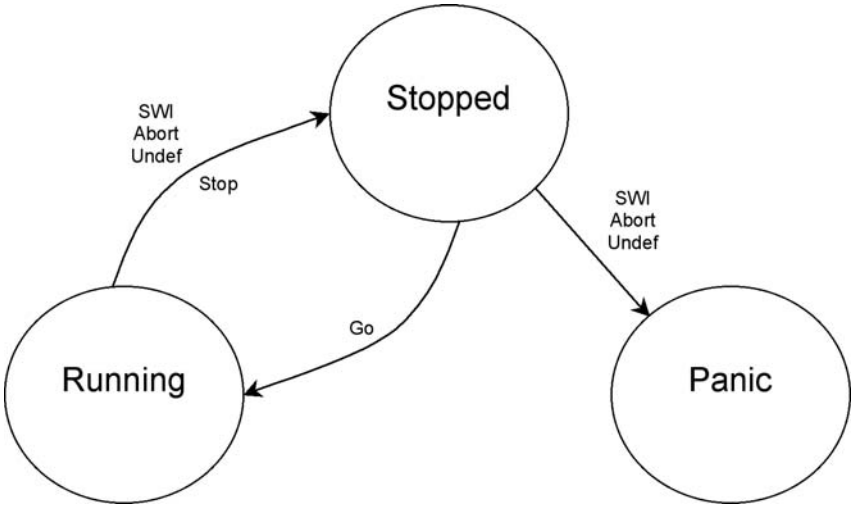
One example of such a product is computer disk drives. Stopping the interrupt-driven control system of a disk drive while the head is in motion usually causes the head to impact the disk, thereby damaging the unit under test. A disk drive designer using the SourcePoint debugger can simply turn RealMonitor on and examine or change values in memory without stopping the target system.

Basic Setup

RealMonitor support consists of three components: one integrated with the SourcePoint debugger on the host computer, one that resides on the emulator, and a target agent that resides on the device under test. The emulator component and the SourcePoint debugger on the host are provided by American Arium. The target agent must be built into your code according to the procedure in *the ARM RMTarget Integration Guide*. Information on the RealMonitor target agent can be found on the ARM website.

Limitations

RealMonitor provides a means for running, stopping, stepping and setting breakpoints in application code while the device under test continues to service interrupts. If a breakpoint, watchpoint, semihosting SWI or other event attempts to stop the processor while the application is stopped, RealMonitor will enter its Panic state. There is no provision to allow nested events to be saved and restored. So, for example, if your application has stopped at one breakpoint, and another breakpoint occurs in an IRQ handler, RealMonitor enters the Panic state. No debugging can be performed after RealMonitor enters this state. The following state diagram shows the operation of RealMonitor.



SourcePoint Support

SourcePoint provides the user with most of the RealMonitor supported functionality. While the interrupt handler continues to run, the foreground application may be started, stopped and single-stepped. The user can also examine and modify processor registers while the foreground application is stopped.

RealMonitor capabilities that SourcePoint does not support at the time of this writing include the following:

- SWI semihosting
- Application or RTOS-specific data logging
- Examining and modifying coprocessor registers
- Sampling of the Program Counter

These are not technical limitations, and some of these functions may be added at a later date.

RealMonitor™ Support and Program Caching in SourcePoint™

Using RealMonitor in SourcePoint

In order to use RealMonitor, the user must put the target into Monitor mode. To do this in SourcePoint, the monitor-mode variable must be set to one. Conversely, Monitor mode is disabled by setting the monitormode variable to zero.

Enabling Monitor Mode can be done at the Command window prompt with the command:

monitormode = 1

If this feature is used on a regular basis, using one of the User-Defined Macro buttons on the toolbar to implement the command will save the effort of entering the command in the Command window.

Caution: Monitor mode can be enabled only when the processor is running.

To ensure that the target is in Monitor mode, check the right-most portion of the status bar found at the bottom of the SourcePoint screen. If the target is in Monitor mode, it will be indicated like this:

Diagram 1

Once in Monitor Mode, the processor continues servicing interrupts when the target is “stopped”. When the processor is “stopped” in Monitor mode, all of the functions normally available the stopped state in Halt mode are available, but critical interrupts are still serviced. In addition, once in Monitor Mode, memory may be read or written while the target is in the running state.

Reading/Writing Memory

Diagram 2 shows the Memory window from an actual debugging session. Notice that the status bar at the bottom of the screen indicates that the processor is running. The green values in the memory window are those that have been modified by the user typing in the new values. Memory contents may be read and written from the Command window as well.

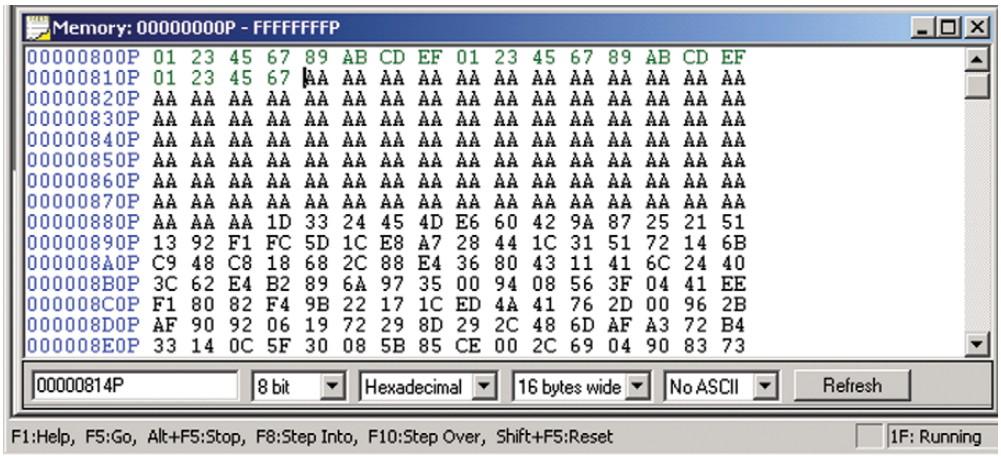


Diagram 2

Start/Stop Trace

Disassembling ETM trace requires the opcodes for the addresses emitted by the ETM. Normally the opcode information would be read from the memory of the target system once the target was stopped. With Realmonitors ability to read target memory without stopping the target, ETM trace can be disassembled without stopping the target system.

Diagrams 3 and 4 are screen captures of the Trace window. Note the right-most button on the dialog bar of the Trace window. This button toggles between “Start Trace” and “Stop Trace”. Also note that the status bar at the bottom of both screens indicates that the processor is running. The ability to start and stop trace, then disassemble the trace while the processor is running, is one of the most powerful features of RealMonitor.

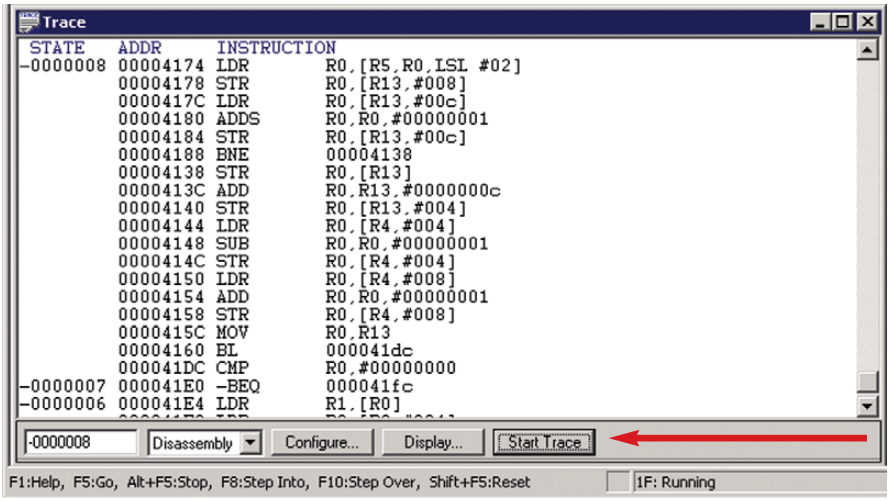


Diagram 3

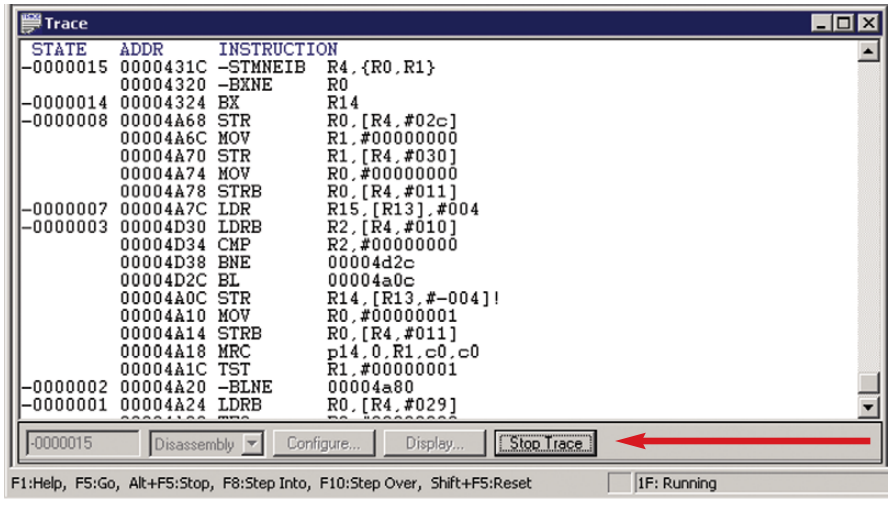


Diagram 4

Omitting RealMonitor code from trace

While capturing trace in Monitor mode, it is highly desirable to filter RealMonitor code from the trace so that only the code under test is shown. This maximizes user of the trace buffer depth, and makes analysis of the trace much easier.

SourcePoint versions prior to release 6.7.2 used a dedicated option in the Emulator tab in the Preferences dialog to control the Realmonitor address range to be omitted from instruction tracing. See the diagram 5. This worked well as long as the user was aware of this feature and the address range of the Realmonitor code never changed.

Starting with SourcePoint release 6.7.2, this GUI feature has been removed. The desired results can still be achieved using an ETM range breakpoint and sequence to exclude instruction tracing for the Realmonitor address range.

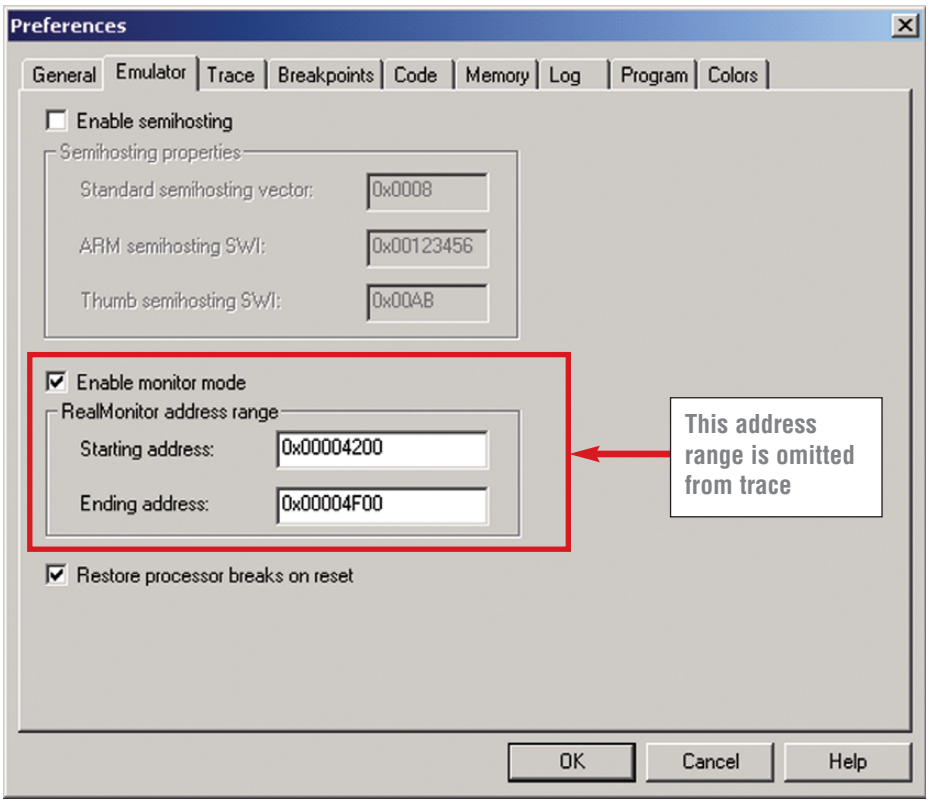


Diagram 5

Program Caching

There are times during the debugging phase of a project when RealMonitor functionality is desired but has not been provided ahead of time, or is not supported by the silicon. The SourcePoint debugger provides some of the same functions without the target is running RealMonitor code.

Normally, the Trace and Code windows are not viewable or controllable while the processor is running. If the processor is not in Monitor Mode, it is simply not possible to access target memory in order to refresh the information in these, or any other memory-based windows without stopping the target system. The SourcePoint debugger gets around this limitation by caching a copy of the program's code image.

Enabling/Disabling Program Caching

Program caching is enabled or disabled using the context-sensitive menu in the Trace or Code window. The context-sensitive menu is found in the menu bar for the focus window. Alternatively, the context-sensitive menu for any window can be accessed by right-clicking at an arbitrary location in the desired window.

From the resulting menu, hold the mouse pointer over "Disassembly Uses…" The resulting choices are *Target Memory and Cached Program*. Choose *Target Memory* to disable program caching. Choose *Cached Program* to enable program caching. The Disassembly Uses settings for the Code and Trace window are independent of each other. Set each one as desired.

Start/Stop Trace

When program caching is enabled, and the processor is running, a "Stop Trace" button is enabled on the dialog bar of the Trace window. Clicking this button causes trace collection to stop without stopping the target system. When program caching is enabled, the target is running, and the trace is stopped, the "Stop Trace" button becomes a "Start Trace" button. Clicking this button restarts trace collection.

Using the ETM Trigger

The ETM provides a complex triggering mechanism. When tracing is stopped, triggers can be added or edited in the SourcePoint Breakpoints window. The ability to do this while the processor is running is only possible when program caching is enabled. RealMonitor does not provide this functionality.

In order for a breakpoint to cause tracing to stop without stopping the target, it must be an ETM breakpoint, and not a Processor or Software breakpoint. A Sequence can then be added in the lower half of the Breakpoints window to stop trace.

In order to cause an ETM breakpoint to start or stop trace, click the Add button at the bottom of the Breakpoints window (see diagram 8). In the Add/Edit Sequence dialog box a check box labeled "stop target" appears when the "trigger" action is selected (only when program caching is enabled). When this box is checked, an ETM trigger stops the processor. In other words, enabling the option causes an ETM trigger to function as a breakpoint which stops both the target and tracing. When this box is unchecked, an ETM trigger simply stops the collection of trace and refreshes the trace buffer. See diagrams 6 and 7.

RealMonitor™ Support and Program Caching in SourcePoint™

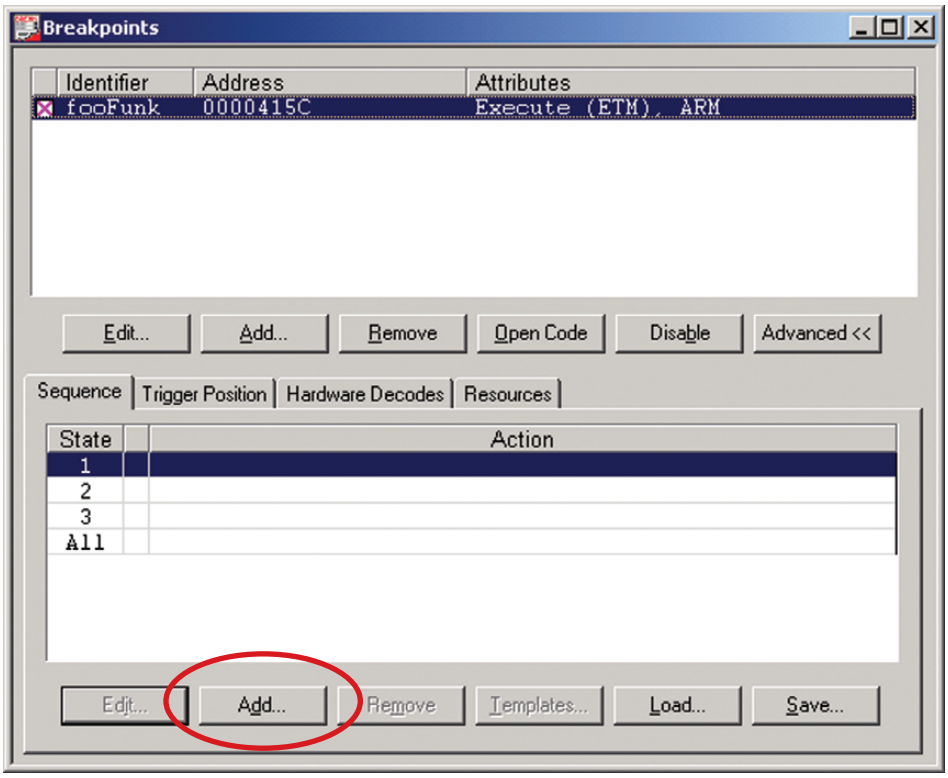


Diagram 6

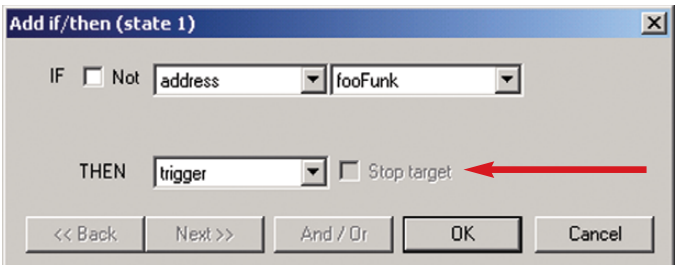


Diagram 7

Limitations

There are certain limitations to consider when using program caching in SourcePoint. While a powerful tool for debugging most application code, it does not track the changes in self-modifying code, and, therefore, should not be used if the user's unit under test runs any such code. Similarly, the cached code remains unaltered if a stray pointer or some other error causes the code on the unit under test to be overwritten. Most users find the advantages of program caching to outweigh these limitations much of the time.

14811 Myford Road, Tustin, CA 92780 Voice: 877-508-3970 in the US or 714-731-1661 outside the US Fax: 714-731-6344 E-mail: info@arium.com
Web www.arium.com

A0149A