

Linux Debug using SourcePoint™

Arium offers an outstanding debug solution for Linux that delivers seamless debug of the kernel, device drivers, and applications with one set of tools. Users can directly launch from or attach to any task from the debugger interface. Processes and their states are visible in a single table; users can access associated source code simply by clicking on a listed process. Shared libraries can be debugged.

Visibility is immediate from bring-up or reset. A console window is available through the JTAG. No more waiting until a serial or ethernet port is "live". No more waiting for the kernel and console drivers to come up. No more waiting, period.

Arium's hardware-assisted SourcePoint™ debugger is the company's flagship software debugger for Intel® Atom™ and ARM7/9/11/Cortex™, XScale, and TI OMAP™ processors. The tool, versatile, customizable, and reliable, offers excellent visibility to c and cc+ and its execution.

Features and Benefits

- Kernel debug straight out of reset - an industry first!
- Dynamic debug of loaded Linux kernel modules - another first!
- Application and process debugging with Linux OS awareness for a complete debug solution in one package.
- Linux console output through JTAG port eliminating serial or ethernet port requirement
- Stack trace viewing for quick context information.

The screenshot displays the SourcePoint debugger interface. At the top, the 'Operating System Resources' window shows a table of processes:

PID	PPID	TTY	Uid	Size	State	Command
8	1	?	root	0k	Sleep	[jffs_gcd]
41	1	ttyDCC	root	872k	Sleep	/bin/sh
42	1	ttyDCC1	root	872k	Sleep	/bin/sh
43	1	ttyDCC2	root	872k	Sleep	/bin/sh
67	1	ttyDCC2	root	80k	Sleep	/home/gdbserver
68	67	ttyDCC2	root	848k	Trace	/bin/busybox
69	43					

Below the table, the 'Code Process[/bin/busybox]:68*' window shows source code for the selected process. The current line of execution is highlighted at line 249:

```

241 #ifdef BB_FEATURE_AUTOWIDTH
242     ioctl(fileno(stdout), TIOCGWINSZ, &win);
243     if (win.ws_col > 0)
244         terminal_width = win.ws_col - 1;
245 #endif
246
247     printf("%5s %-7s %-8s %6s %5s %s\n", "PID", "
248         "Size", "State", "Command");
249     while ((entry = readdir(dir)) != NULL) {
250         if (!isdigit(*entry->d_name))
251
252
253
254
255
    
```

To the right of the code, a 'Symbols - Stack Process[/bin/busybox]:68*' window displays the current stack trace:

StackFrame	Value
ps_main(argc 0x00000001, argv 0xBFFFFFF4)	
printf_unknown+118	
busybox_main(argc 0x00000001, argv 0xBFFFFFF4)	
show_usage()	
concat_path_file(path 0xBFFFFFF4 "%s", filename 0x000561C0 "")	
busybox_main(argc 0x00000002, argv 0xBFFFFFF4)	
__assert_fail+88	

At the bottom left, the 'Tasks' window shows the address 000425F0.

Source level debug of a Linux task, including the entire stack

The Linux OS-aware features of SourcePoint reflect a number of important capabilities for users who are working on Linux-based embedded systems. These include:

- Full symbolic, source-level debugging of Linux kernel code, including loadable kernel modules.
- Source-level debugging of Linux embedded applications (including shared libraries), the ability to start or stop a Linux process, attach to a process, view source and symbols for a process, and set breakpoints within a process.
- Launch of or attachment to processes with seamless transitions to and from the kernel and each process.
- Specialized breakpoints to stop the execution of a process without stopping the processor or causing it to enter debug mode.
- Flash programming for kernel and file system download.
- Linux console hosting devices from within SourcePoint, eliminating the need for a serial port or video device on the target and simplifying the debugging of "headless" systems.



Linux Debug using SourcePoint™

Managing Run Control

The key to a successful run control debug strategy lies in the ability to set accurate breakpoints and step through code. SourcePoint offers processor breaks and unlimited soft breaks via intuitive GUIs. Breaks can also be set from the Code window or a command line.

SourcePoint uses the usual stepping commands along with Go and Halt to step through source or assembly-level code. SourcePoint's C-like command language includes not only industry standard run control commands, but lets the developer execute loops, use data and array variables, access file I/O, and more. Unlike some command languages, this one does not require the developer to know a two-letter code for each command.

Well designed windows can be opened to view the state of the processor and make modifications to values, including Symbols windows, Registers windows, Memory windows, and user-defined Watch windows. A change in a value made in one of these kinds of windows is updated automatically in others. In multi-core environments, developers together or individually can mask processors and start, step, and stop those not masked.

Shortcuts Within Shortcuts

SourcePoint incorporates hundreds of options, commands, and functionalities designed to spur the debug process forward. Windows are designed to be intuitive. They can be docked, floated, or minimized. Commands are available from multiple locations - menu bars, icon bars, context menus, a command line. Symbols and their values are easy to find and change.

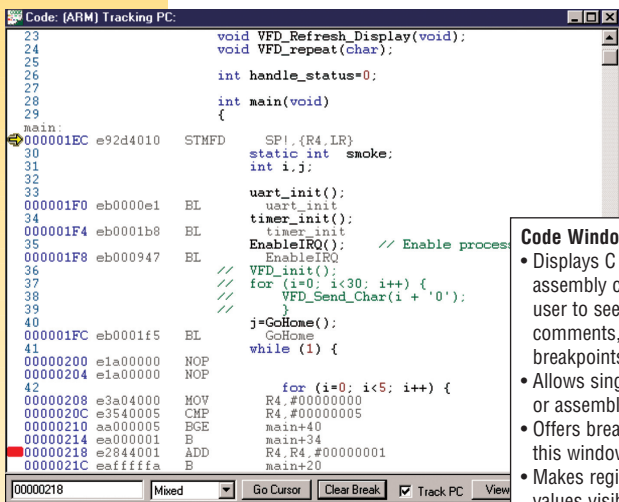
Items are grouped logically in intuitive windows and dialog boxes. For example, target configuration options exist under a single view. From the dialog, the memory map of the target can be defined, the type and address range of flash memory devices declared, and target flash operations performed. Target configurations can be loaded from a user's target database file and saved to SourcePoint and/or the target database file.

SourcePoint offers a number of userdefined options. This includes a window that allows definition of memory-mapped I/O devices and related registers and areas of memory in one view. Users can keep track of multiple devices without having to keep multiple views on their screen.

Outstanding Support

American Arium offers exceptional service and support for all of its debug solutions. Highly qualified technical support staff are available during regular working hours, and delays getting to them are minimal. Often they can pinpoint a problem immediately or on review of a dump of the log and project files. Support staff can also troubleshoot particularly difficult problems via WebEx™, an online, interactive solution that lets them see a developer's code but lets the developer control the session. Additionally, there are downloads and technical documentation available on the Arium Web site.

SourcePoint ships with Arium hardware or separately as an upgrade to previously purchased hardware-assisted solutions. For more information, contact your sales representative or Arium tools distributor or visit our Web site at www.arium.com.



Code Window

- Displays C or C++ source or assembly code, or allows the user to see both; also displays comments, symbols, and breakpoints
- Allows single stepping (in C or assembly code)
- Offers breakpoint setting from this window
- Makes register or variable values visible via flyover help
- Works with multi-processor systems via multiple Code windows

