

# smxAware<sup>TM</sup>

## PRODUCT BRIEF

smxAware adds SMX<sup>®</sup> awareness to popular embedded debuggers produced by other vendors. This is a great convenience during debugging and results in significant time savings. smxAware is a DLL which is loaded on the host by the host portion of the debugger. The following debuggers are currently supported (for Windows hosts):

<u>Debugger</u>	<u>Processor</u>	<u>Vendor</u>
CodeWarrior Professional	ARM, CF, PowerPC	Freescale
EWARM/C-SPY	ARM	IAR Systems
Paradigm C++	x86 (RM, 32-bit PM)	Paradigm Systems
SingleStep	PowerPC	Wind River
Soft-Scope / pmScope	x86 (32-bit PM)	Micro Digital
visionCLICK	CF	Wind River
VisualProbe	x86 (16 & 32-bit PM)	Beacon Development Tools
Visual Studio	smxSim	Microsoft

### Kernel Objects Displayed

- Tasks
- LSRs
- Exchanges
- Messages
- Semaphores
- Mutexes
- Events
- Event Tables
- Ready Queue
- Timers
- Pipes
- Buckets
- Heaps
- Stacks
- Print Statements
- Handle Table
- Diagnostics

### SMX Add-on Module Objects Displayed

- smxFS
  - Disk info
  - File info for open files
- smxNS
  - Net status
  - Buffer status
  - ARP status
  - Route status
- smxUSBD
  - Device controller name
  - Function drivers registered
  - Device, configuration, interface, and endpoint descriptor details
- smxUSBH
  - Host controller name
  - Class drivers registered
  - Plugged-in devices
  - Device, configuration, interface, and endpoint descriptor details

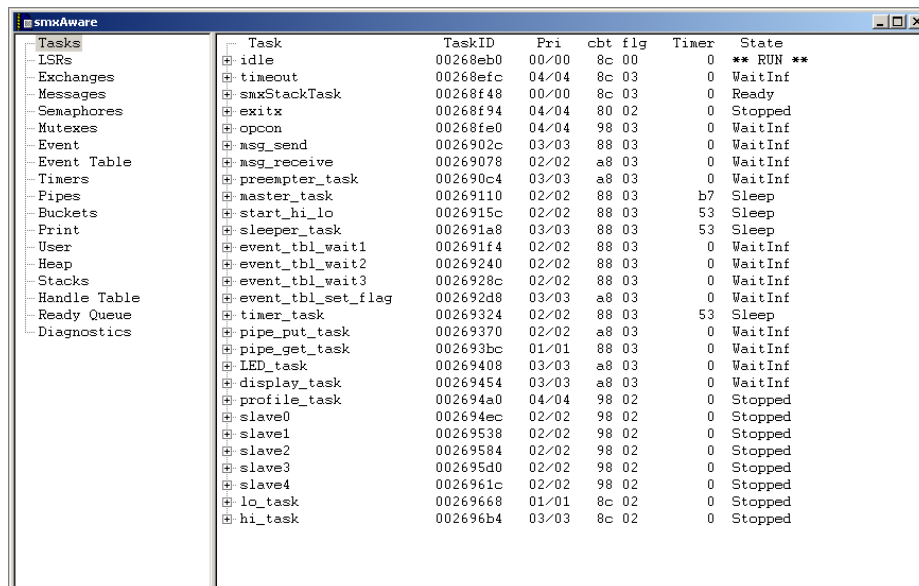
Add-on displays are currently available only for CodeWarrior and IAR ARM little endian.

## Operation

smxAware is a dialog box that opens when selected from the debugger's main menu. It presents a list of the object types listed above. When one is selected, all objects of that type are displayed. A one-line summary is given for each object of that type. In many displays, double-clicking on an entry expands it to show more information. For example, double clicking on a task entry shows all fields of its control block. Double clicking on an exchange entry shows all the messages or tasks waiting at that exchange. All levels of multi-level objects are shown.

smxAware supports task-specific breakpoints for some debuggers.

smxAware for CodeWarrior displays additional information for suspended tasks. Each task has a *thread window* associated with it that shows where the task will resume execution, and the call stack up to that point. It also shows local variables, if any, and the task's registers. This information is determined by smxAware from the task context that was saved on the stack when the task was suspended. For CodeWarrior, the smxAware window is a *non-modal* dialog box meaning that you can watch it while stepping through your code.



The screenshot shows the smxAware dialog box with a tree view on the left and a table of task details on the right. The tree view includes categories like Tasks, ISR's, Exchanges, Messages, Semaphores, Mutexes, Event, Event Table, Timers, Pipes, Buckets, Print, User, Heap, Stacks, Handle Table, Ready Queue, and Diagnostics. The table on the right lists tasks with columns for Task, TaskID, Pri, cbt, flg, Timer, and State.

Task	TaskID	Pri	cbt	flg	Timer	State
-- Task						
-- ISR's						
-- Exchanges						
-- Messages						
-- Semaphores						
-- Mutexes						
-- Event						
-- Event Table						
-- Timers						
-- Pipes						
-- Buckets						
-- Print						
-- User						
-- Heap						
-- Stacks						
-- Handle Table						
-- Ready Queue						
-- Diagnostics						
⊕ idle	00268eb0	00/00	8c	00	0	** RUN **
⊕ timeout	00268efc	04/04	8c	03	0	WaitInf
⊕ smxStackTask	00268f48	00/00	8c	03	0	Ready
⊕ exitx	00268f94	04/04	80	02	0	Stopped
⊕ opcon	00268fe0	04/04	98	03	0	WaitInf
⊕ msg_send	0026902c	03/03	88	03	0	WaitInf
⊕ msg_receive	00269078	02/02	a8	03	0	WaitInf
⊕ preempter_task	002690c4	03/03	a8	03	0	WaitInf
⊕ master_task	00269110	02/02	88	03	b7	Sleep
⊕ start_hi_lo	0026915c	02/02	88	03	53	Sleep
⊕ sleeper_task	002691a8	03/03	88	03	53	Sleep
⊕ event_tbl_wait1	002691f4	02/02	88	03	0	WaitInf
⊕ event_tbl_wait2	00269240	02/02	88	03	0	WaitInf
⊕ event_tbl_wait3	0026928c	02/02	88	03	0	WaitInf
⊕ event_tbl_set_flag	002692d8	03/03	a8	03	0	WaitInf
⊕ timer_task	00269324	02/02	88	03	53	Sleep
⊕ pipe_put_task	00269370	02/02	a8	03	0	WaitInf
⊕ pipe_get_task	002693bc	01/01	88	03	0	WaitInf
⊕ LED_task	00269408	03/03	a8	03	0	WaitInf
⊕ display_task	00269454	03/03	a8	03	0	WaitInf
⊕ profile_task	002694a0	04/04	98	02	0	Stopped
⊕ slave0	002694ec	02/02	98	02	0	Stopped
⊕ slave1	00269538	02/02	98	02	0	Stopped
⊕ slave2	00269584	02/02	98	02	0	Stopped
⊕ slave3	002695d0	02/02	98	02	0	Stopped
⊕ slave4	0026961c	02/02	98	02	0	Stopped
⊕ lo_task	00269668	01/01	8c	02	0	Stopped
⊕ hi_task	002696b4	03/03	8c	02	0	Stopped

Example of Task Display (non-modal version)

## Graphical Analysis Tool

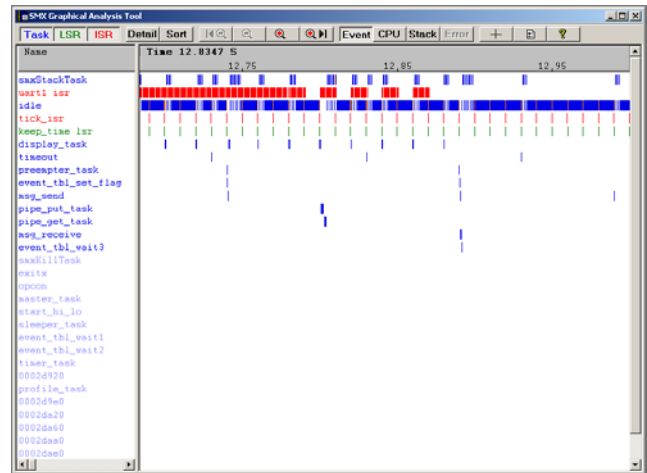
smxAware for CodeWarrior, EWARM/C-SPY, visionCLICK, and Visual Studio includes graphical displays. Below are the main views. We recommend that you download the demo version so you can see it up close ([www.smxrtos.com/demo/gatdemo.zip](http://www.smxrtos.com/demo/gatdemo.zip)). Or download one of the SMX Evaluation Kits that includes it (see [www.smxrtos.com/eval](http://www.smxrtos.com/eval)).

A standalone .exe is also provided to allow you to analyze collected data off-line.

### Event Timelines

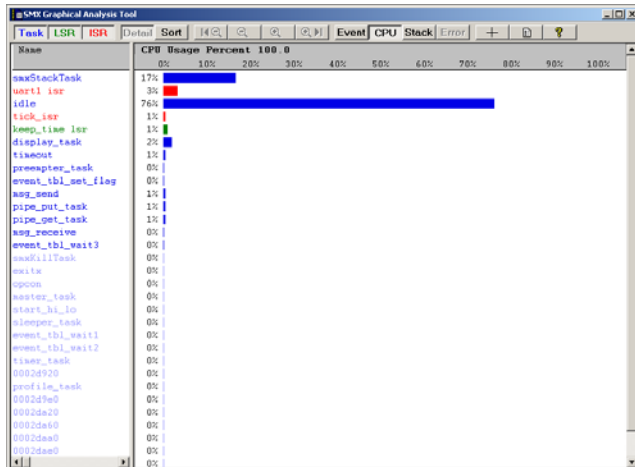
This view shows when tasks, lsr's, and isr's ran. Inside the bars it shows when srr's (smx kernel calls) occurred. User events can also be added to the application. This display can be zoomed to a very fine level of detail. Time resolution can be microseconds, or less, depending upon the capture rate in the target. When the Details button is pressed and the mouse cursor is moved over an event, information about the event appears in a small window next to the cursor. For an smx kernel call, this information includes the name of the call, all parameters, and the return value. Hence, the display is clean and uncluttered, yet it is easy to get detailed information about events.

Event timelines are an important visualization tool needed to understand and debug modern, complex embedded systems. The precise time that each isr, lsr, or task started and stopped is shown, along with the cause. Timelines can be moved or resorted by sequence of occurrence. This is helpful to focus on the few timelines of interest, at the moment. The display can be



zoomed in or out and scrolled left or right and up or down, if there are too many timelines to fit the screen. Line thickness can be reduced to get more on the screen. isr's, lsr's, and tasks can be selectively turned off or on to create more screen space. Prior to zooming, a reference point can be set, which stays still while zooming in or out. These, and other features, make the smxAware timeline display convenient and effective to use.

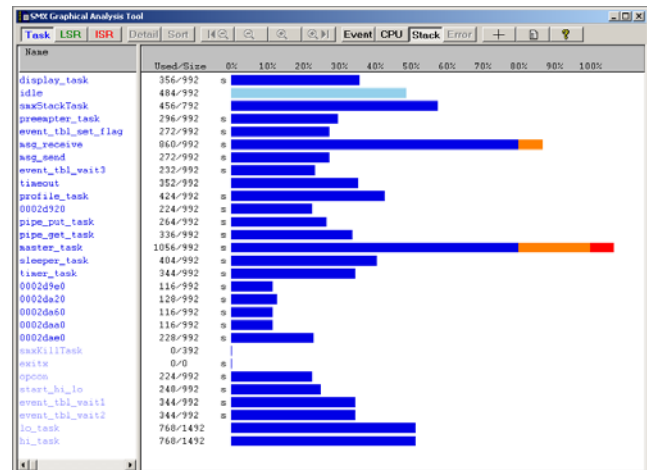
## CPU Usage



This view shows how much time each task, isr, and isr ran during the sample period, as a percent of the sample period. Known as “task profiling”, this window helps to resolve performance issues. (Note that in the diagram to the left the idle task, which has lowest priority, is running 76% of the time. Hence there is plenty of bandwidth for useful work in this system.)

## Stack Usage

This view shows how much stack each task uses. Orange indicates stacks that are close to their maximum allocated sizes. Red indicates stacks that have overflowed. To the left of each bar is the actual number of bytes used/stack size. The bars show percentages. Stacks can overflow without bringing the system to a halt because in debug mode, smx allocates extra space for each stack. Without this display, stack overflow is often very difficult to debug in a multitasking system.



All of the above views can be brought up whenever the target is stopped at a break point. On-screen help is also provided.

smxAware, like other Micro Digital products, is sold on a 30-day free trial basis and includes 90 days of support. See the smxAware User’s Guide for more information.