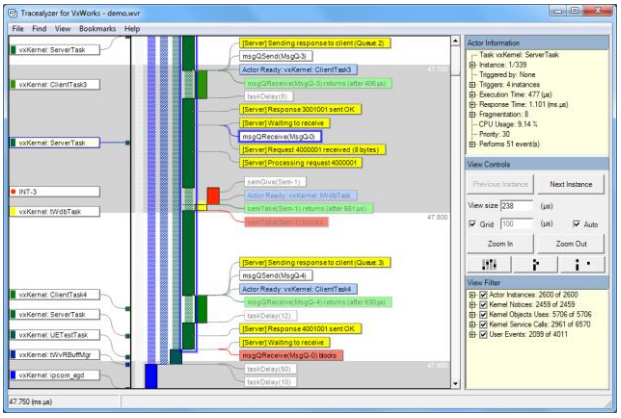
percepio
SENSING SOFTWARE

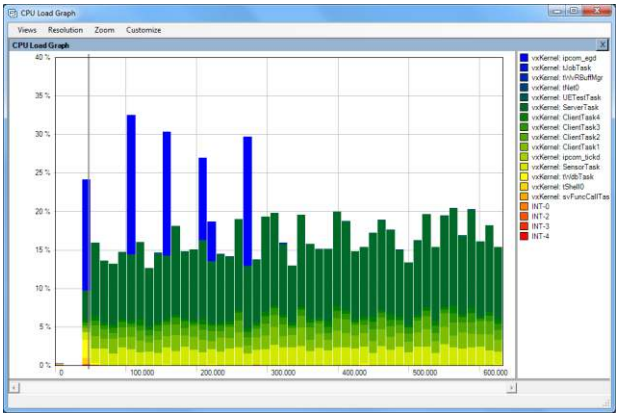
Tracealyzer for VxWorks – Overview and Getting Started Guide

Tracealyzer for VxWorks is a trace visualization tool for VxWorks systems. Tracealyzer gives developers an unprecedented insight into the runtime behavior, which allows for reduced troubleshooting time and improved software quality, performance and reliability. This sophisticated tool offers 20+ views of the runtime world, cleverly interconnected.

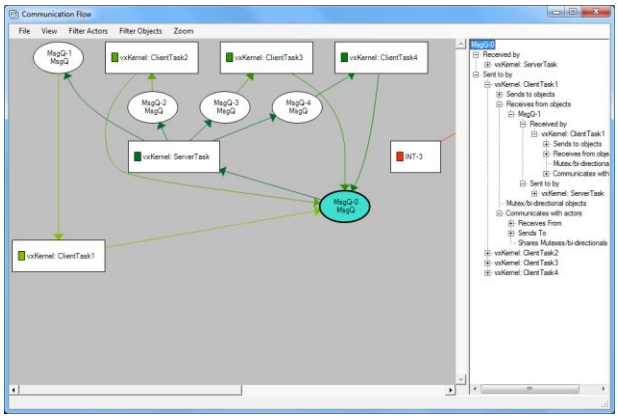
Tracealyzer does not depend on special trace hardware, which means that it can be used in field testing and deployed systems to capture rare errors, otherwise hard to reproduce.



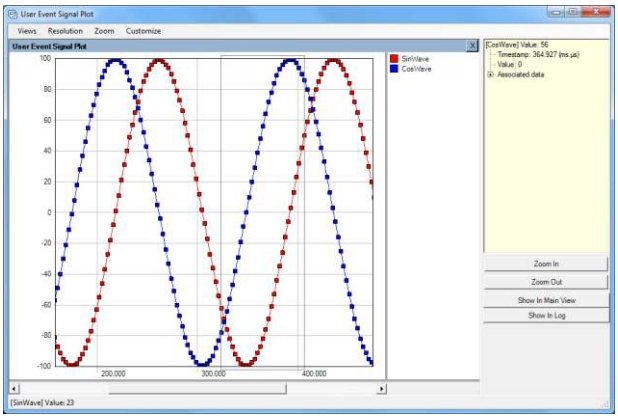
Task scheduling, ISRs and selected events



Visualize CPU load and memory usage



Task and ISR communication flow



Plot important application variables

Tracealyzer contains several advanced analyses that help you faster comprehend the trace data. For instance, it connects related events, which allows you to follow messages between tasks and to find the event that triggers a particular task instance. Moreover, it provides several higher level views such as the Communication Flow graph and the CPU Load Graph, which make it easier to find anomalies in a trace. For detailed information on features offered, we refer to the Users' Manual (http://percepio.com/docs/VxWorks/manual/).

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN

www.percepio.com

Installing Tracealyzer on Windows

Run the provided installer to unpack the application files.

Installing Tracealyzer on Linux

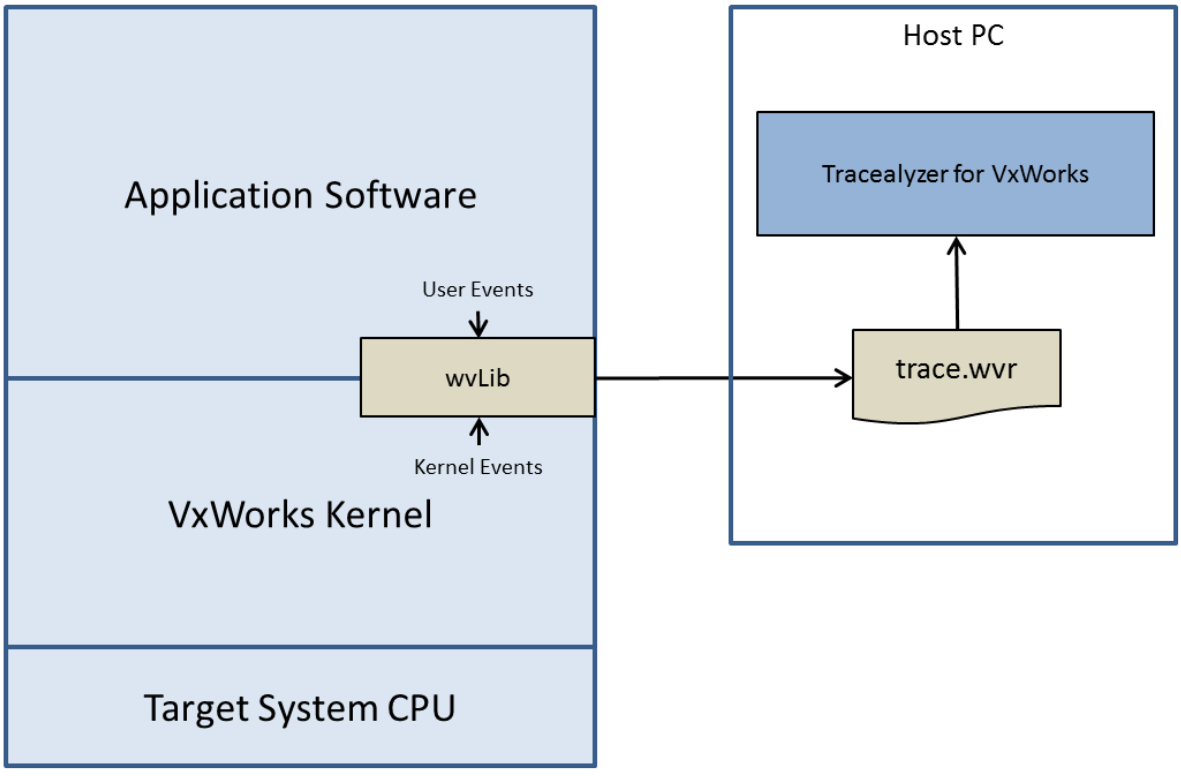
Run the installer program or extract the compressed .tgz or .rpm file to any directory.

Tracealyzer is a .NET application but supports Linux hosts using Mono, an open-source .NET framework. Make sure you have Mono (version 3.8 or newer). If you need to install or update your Mono version, this is found at http://www.mono-project.com.

To activate the evaluation license on Linux, there might be an issue related to SSL root certificates. Mono does not trust any SSL root certificates by default, so we recommend that you use a tool called *mozroots*, included with Mono, to import all root certificates trusted by Firefox into Mono's trusted certificate store. See the Mono FAQ page for more information.

Using Tracealyzer with VxWorks

Tracealyzer for VxWorks visualize trace files in .wvr format from the existing trace recorder in VxWorks (wvLib). Such traces contain kernel events like context-switches and kernel calls, but may also include "User Event" logging from the application code.



The VxWorks recorder stores the trace in a RAM buffer and supports three upload modes:

- Deferred Upload: The trace data is kept in the RAM buffer until an upload command is received. The recording length is limited by the RAM buffer size, but continuous recording is possible by selecting the ring-buffer mode, where the oldest data is overwritten. Another possibility is to stop the recording when the buffer gets full.

- Continuous Upload: The trace data is continuously uploaded from the RAM buffer to the host computer, allowing for essentially unlimited trace length. This is however more intrusive as the periodic uploading puts additional load on the target system.

- Post-Mortem Upload: allows for analyzing the trace leading up to a hard crash. This uses a ring-buffer stored in a persistent memory region, i.e., that is not reset when rebooting.

To make a .wvr trace of your own VxWorks system, you first need to make sure that your VxWorks Image project has been configured to support Wind River System Viewer and that is uses a high-resolution timestamp driver.

The VxWorks trace recorder (wvLib) can be controlled programmatically from the VxWorks application or from System Viewer in Wind River Workbench. For further information, we refer to Wind River System Viewer Users' Guide and the VxWorks documentation for wvLib.

If you have Wind River Workbench installed, you can easily make a recording using System Viewer. The resulting .wvr recording file is usually found in the Workbench project directory. To view them in Tracealyzer, select *File->Open* or simply drag the .wvr file to the Tracealyzer main window.

You may also control the trace recording from your VxWorks application code and thereby also from the target shell. Examples are found in the VxWorks documentation for wvLib.

Note: Integrated functions for VxWorks trace control is planned for the next version (v2.8) of Tracealyzer for VxWorks.

Why use Tracelyzer instead of System Viewer?

Tracealyzer for VxWorks uses the same data source as System Viewer, but provides a more powerful visualization featuring over 20 different views that all are interconnected in clever ways. Tracealyzer is more intuitive to use and provides a much better understanding.

Smart trace visualization: The main trace view uses a vertical time-line and offer several options for rendering the task trace. Events are shown as horizontal color-coded text labels. As shown in Figure 1, the labels automatically spread out to use the available space without overlapping, and can be filtered in several different ways to focus the view (Figure 2).

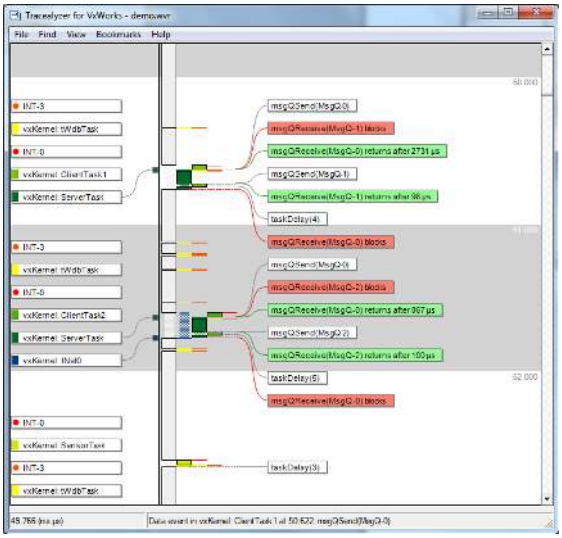


Figure 1: Event labels in main trace view

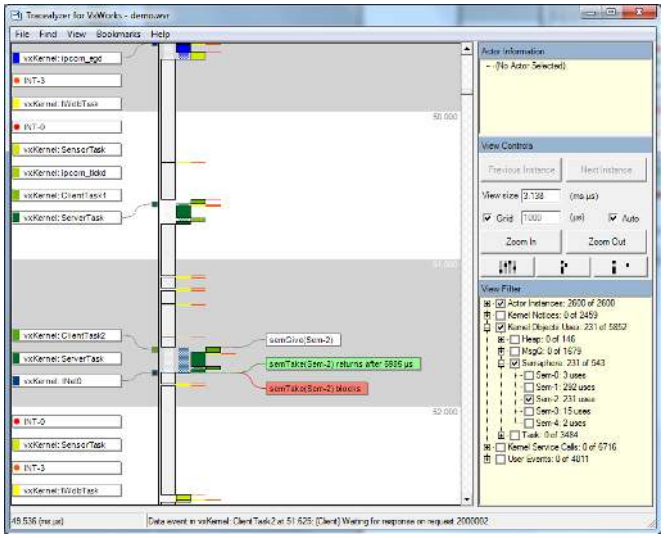


Figure 2: Main view, with event filter

Visualize related events: Dependencies between system calls, tasks and other kernel objects are understood and highlighted by Tracealyzer. As shown in Figure 3, if selecting a blocking event (red label) the corresponding resume event (green label) is highlighted. And if selecting a "msgQSend" call, the matching "msgQReceive" call can easily be found. Moreover, the Communication Flow graph (Figure 4) shows you a dependency graph of the task interactions. This is a high-level view of your system's runtime architecture!

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN

www.percepio.com

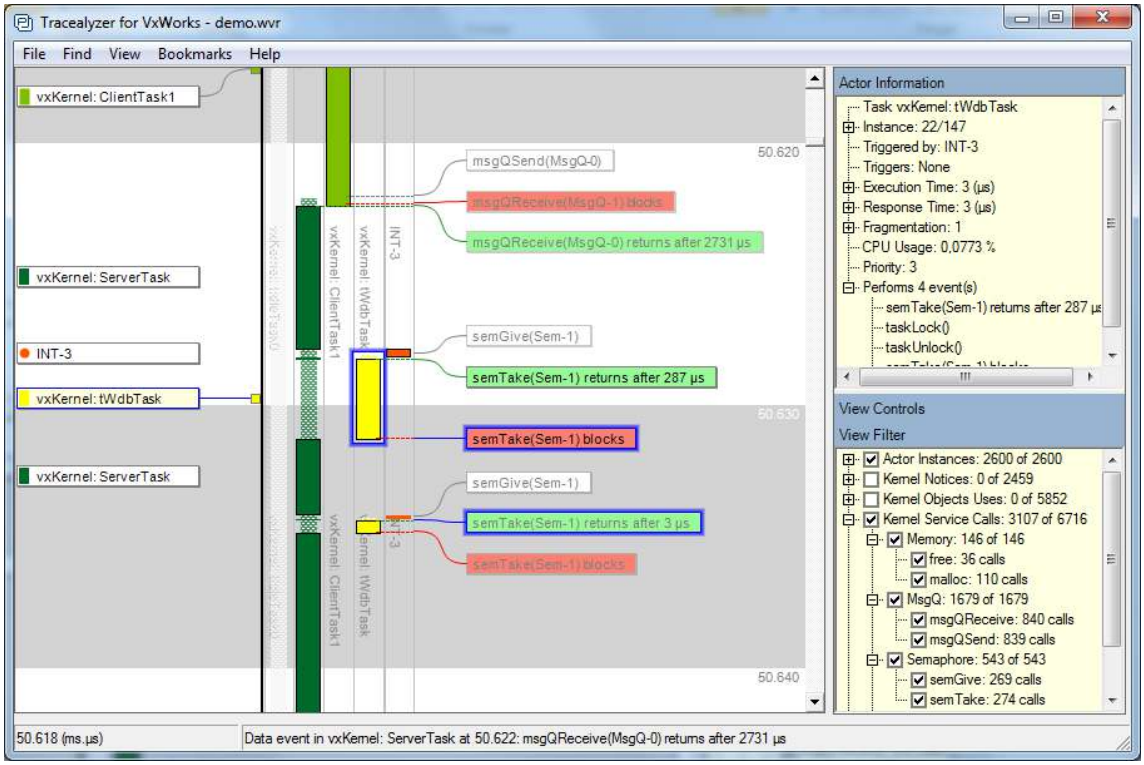


Figure 3: Blocking and resume event highlighted.

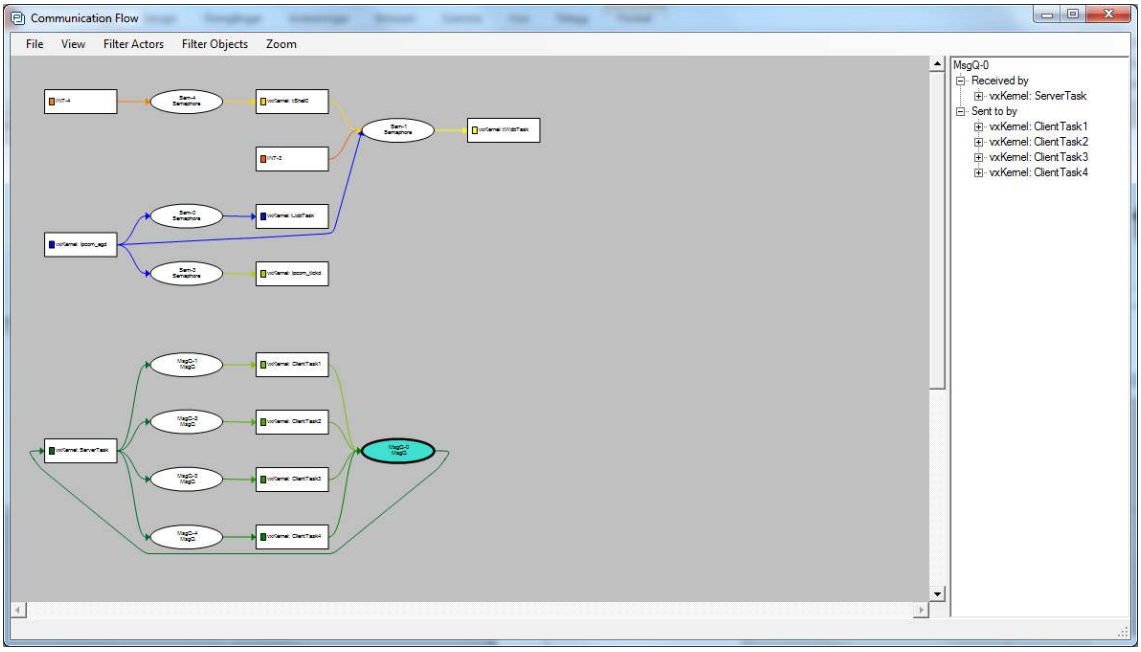


Figure 4: Communication Flow graph showing runtime dependencies between tasks and queue, etc.

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN

www.percepio.com

Integrated data plotting: Tracealyzer allows you to log any application data or event as "User Events", shown as yellow labels. Any data arguments can be plotted, as shown in Figure 5, and the clicking on a data point highlights the corresponding User Event in the main trace view (Figure 6) which allows for correlating the data with task scheduling, interrupts, system calls and other events.

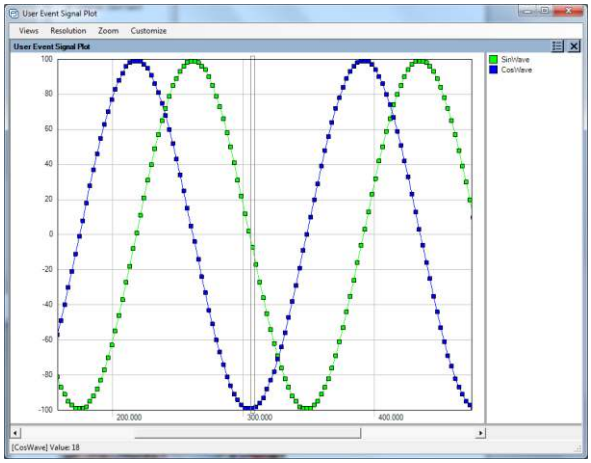


Figure 5: User Event Signal Plot

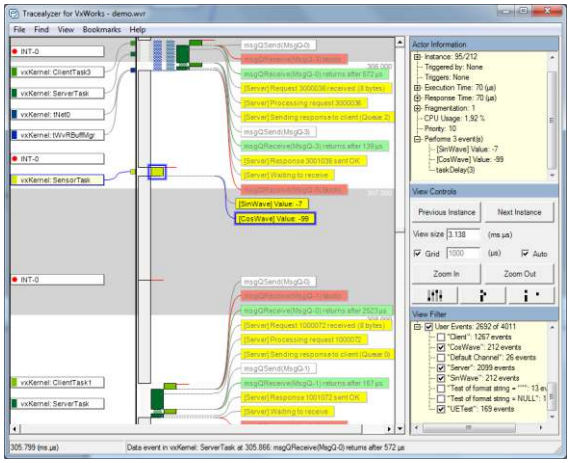


Figure 6: User Events in Main Trace View

Integrated memory usage view: Tracealyzer provides a graph showing dynamic memory allocation over time, i.e., malloc() and free(), as illustrated by Figure 7. This allows you to spot memory leaks and excessive memory usage. Figure 8 shows a more detailed memory allocation view, where the addresses are shown and malloc() calls can be matched against free() calls. Since both memory views are connected to the main trace view, you can easily find and analyze the context of any issues found.

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN
www.percepio.com

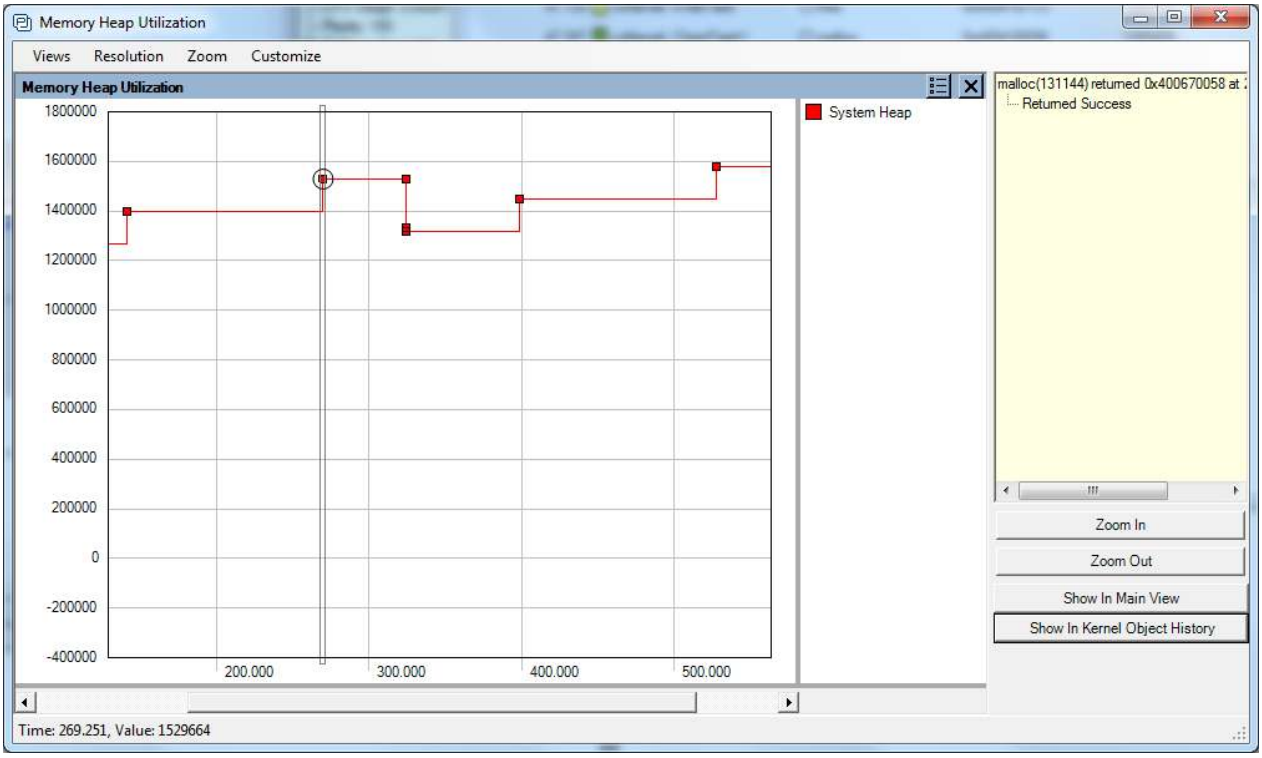
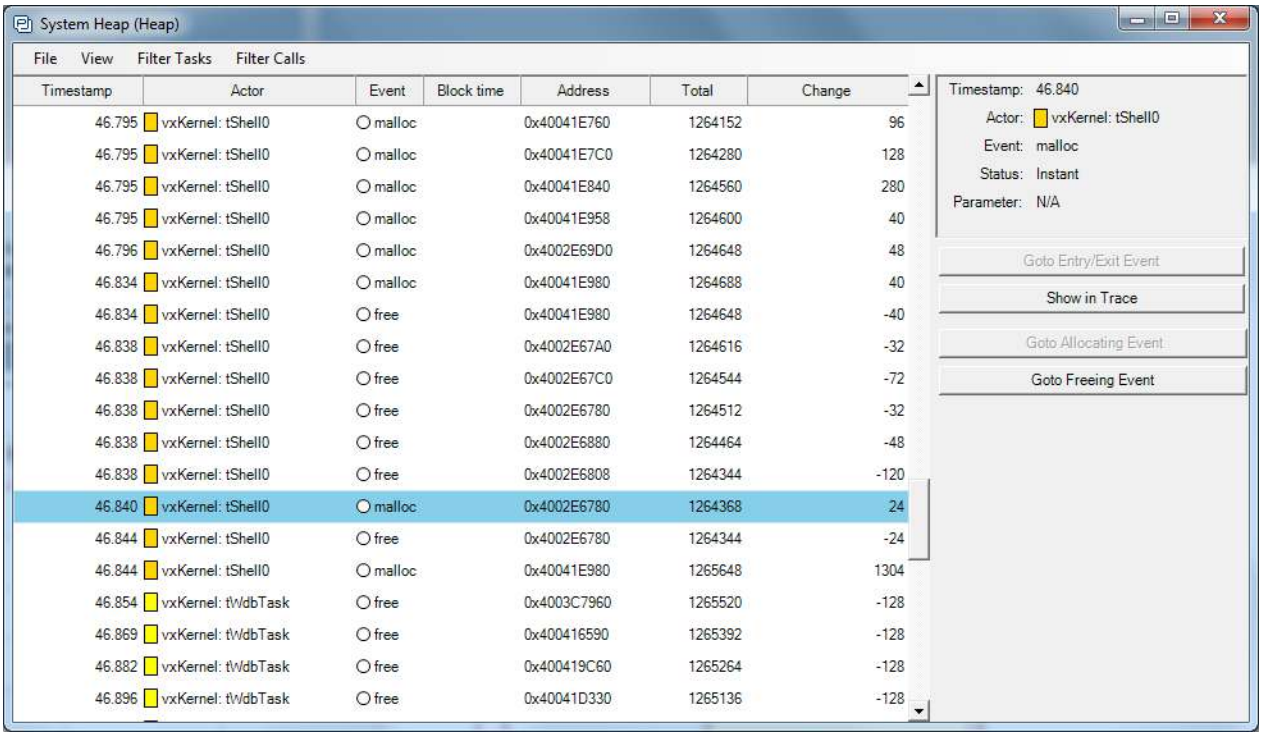


Figure 7: Dynamic memory allocation over time.



Timestamp	Actor	Event	Block time	Address	Total	Change
46.795	vxKernel: tShell0	malloc		0x40041E760	1264152	96
46.795	vxKernel: tShell0	malloc		0x40041E7C0	1264280	128
46.795	vxKernel: tShell0	malloc		0x40041E840	1264560	280
46.795	vxKernel: tShell0	malloc		0x40041E958	1264600	40
46.796	vxKernel: tShell0	malloc		0x4002E69D0	1264648	48
46.834	vxKernel: tShell0	malloc		0x40041E980	1264688	40
46.834	vxKernel: tShell0	free		0x40041E980	1264648	-40
46.838	vxKernel: tShell0	free		0x4002E67A0	1264616	-32
46.838	vxKernel: tShell0	free		0x4002E67C0	1264544	-72
46.838	vxKernel: tShell0	free		0x4002E6780	1264512	-32
46.838	vxKernel: tShell0	free		0x4002E6880	1264464	-48
46.838	vxKernel: tShell0	free		0x4002E6808	1264344	-120
46.840	vxKernel: tShell0	malloc		0x4002E6780	1264368	24
46.844	vxKernel: tShell0	free		0x4002E6780	1264344	-24
46.844	vxKernel: tShell0	malloc		0x40041E980	1265648	1304
46.854	vxKernel: tWdbTask	free		0x4003C7960	1265520	-128
46.869	vxKernel: tWdbTask	free		0x400416590	1265392	-128
46.882	vxKernel: tWdbTask	free		0x400419C60	1265264	-128
46.896	vxKernel: tWdbTask	free		0x40041D330	1265136	-128

Figure 8: Detailed view of memory allocation.

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN

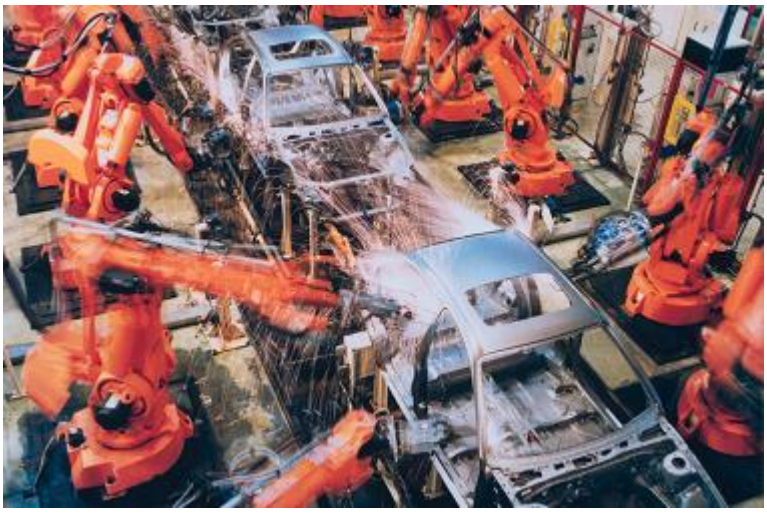
www.percepio.com

Public Reference Customers

Percepio AB has been in business since 2009 and has very satisfied customers all over the world. Below are two Percepio customers who have provided an official public statement.

Jet fighter SAAB JAS-39 Gripen

"The many system views of the Tracealyzer from Percepio made it easy to quickly find solutions that we have not seen using (Wind River) System Viewer. The visualization has several advantages over the System Viewer and makes it much easier to understand the system behavior."
Johan Fredriksson, Software Architect, SAAB AB.



Industrial robots from ABB

"ABB Robotics is using the first generation Tracealyzer in all of the IRC5 robot controllers shipped since 2005. The tool has proven its value many times in all corners of the world."
Roger Kulläng, Global System Architect, ABB Robotics.

For further information contact support@percepio.com or call us at +46 21 146 210 between 8.00-16.00 Central European Time. We are happy to offer an online demonstration.

Percepio AB
Köpmangatan 1A
72215 Västerås
SWEDEN
www.percepio.com