ERIKA Enterprise pre-built Virtual Machine for Freescale PPC

Including Lauterbach TRACE32

Version: 1.2

September 18, 2014

About Evidence S.r.l.

Evidence is a company operating in the field of software for embedded real-time systems. It started in 2002 as a spin-off company of the Real-Time Systems (ReTiS) Lab of the Scuola Superiore Sant'Anna (Pisa, Italy). Today, Evidence is a dynamic company having collaborations in the field of electronics, telecommunications, automotives, and industrial automation.

People at Evidence are experts in the domain of embedded and real-time systems, with a deep knowledge on the design and specification flow of embedded software, especially for the embedded market.

Besides providing consultancy services, Evidence also provides: BSPs based on Linux for embedded devices, evaluation boards featuring most innovative 8, 16 and 32-bit microcontrollers for the embedded market, development tools for making embedded software development easier, and tools for the schedulability analysis of real-time tasks running on your final product.

For more information see: http://www.evidence.eu.com

Contact Info

Evidence Srl,
Via Carducci 56
Località Ghezzano
56010 S.Giuliano Terme
PISA Italy

Tel: +39 050 99 11 224
Fax: +39 050 99 10 812

For more information about Evidence products, please send an e-mail to the following address: info@evidence.eu.com. Other information about the Evidence product line can be found at the Evidence web site at: http://www.evidence.eu.com.

Contents

List of Figures

About this document

This document describes the installation, first setup and first demo run of the ERIKA Enterprise pre-built Virtual Machine for Freescale PowerPC. This Virtual Machine was done in collaboration with Lauterbach, who provided the TRACE32 Instriction set simulator to run ERIKA applications.

Function of the document

The function of this document is to provide a quick start guide for using the Virtual Machine with a demo example.

Document history

Version	Date	Author	Company	Change Description
1.0	June 2014	Paolo Gai	Evidence Srl	Initial version
1.1	June 2014	Paolo Gai, Marco Ferrario	Evidence Srl, Lauterbach Srl	Typos
1.2	Sept 2014	Paolo Gai	Evidence Srl	Added support for the GCC PPC crosscompiler

Acronyms

Acronym	Meaning
`TRACE32`	Lauterbach TRACE32.
`ISS`	Instruction Set Simulator

1 Introduction

Installing a complete development and debugging environment for automotive microcontrollers always involves a lot of integration work between compilers, debuggers, development environments, makefiles, and so on. This Virtual Machine is aimed at providing a quick solution for all these problems, providing a Linux platform with all software preinstalled and ready to work, allowing you to run OSEK/VDX applications on a simulated hardware platform.

In other words, this Virtual Machine provides a complete virtual environment where you will be able to:

- Edit your OSEK/VDX application using the open-source OSEK/VDX Kernel ERIKA Enterprise;
- Compile your application using the pre-installed GCC crosscompiler for PPC;
- Debug, Trace and Run a target application on a simulated Freescale PPC MPC5674F Mamba using the pre-installed Lauterbach TRACE32 Instruction Set Simulator.

1.1 Requirements

This Virtual Machine can be run easily on VirtualBox (`https://www.virtualbox.org/`). VMware Player can also be used, although we id not try yet.

No additional hardware or evaluation board is required to run the code compiled, as everything can be run on top of the Lauterbach TRACE32 Instruction Set Simulator (ISS).

1.2 Licensing

The Virtual Machine described in this document includes various open-source and non-open-source software. The following points shortly describe the main licenses of the tools which has been integrated during this work:

- The Linux Distribution is a standard Ubuntu distribution. For more information about Ubuntu and the software licenses included in this Linux distribution please refer to the following website: `http://www.ubuntu.com/`.
- ERIKA Enterprise is distributed mainly under the GPL2+Linking Exception License (`http://en.wikipedia.org/wiki/GPL_linking_exception`), plus other licenses used for user libraries in `contrib` directory.

- Eclipse, EMF, and the RT-Druid plugins are distributed under the EPL License (http://en.wikipedia.org/wiki/Eclipse_Public_License).
- The GCC crosscompiler for PPC is an open-source project under the GPL license. We integrated the toolchain available at http://www.macraigor.com/full_gnu.htm.
- Lauterbach TRACE32 is proprietary software produced by Lauterbach Gmbh (http://www.lauterbach.com).

1.3 Feedback, bugs, and additional examples

We care about your feedback! Information, feedback, and new demos about ERIKA Enterprise can be provided directly on the ERIKA Enterprise website:

http://erika.tuxfamily.org

For commercial technical support, sales, pricing, order status, and general information and feedback, please contact Evidence Srl directly at the address and phone numbers available at the following web page:

http://www.evidence.eu.com/en/contact-us.html

2 Installing VirtualBox and creating the Virtual Machine settings

2.1 Installing VirtualBox

VirtualBox can be freely downloaded and used also for commercial use from the following website:

https://www.virtualbox.org/wiki/Downloads

All you need to do is to download the VirtualBox installer, and install it on your PC. All the following screenshots will refer to the usage of VirtualBox on a Windows 7 Host machine.

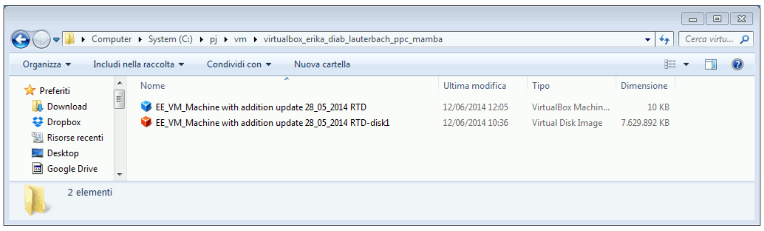
2.2 Downloading the Virtual Machine

The Virtual Machine can be downloaded from the following website:

http://www.erika-enterprise.com

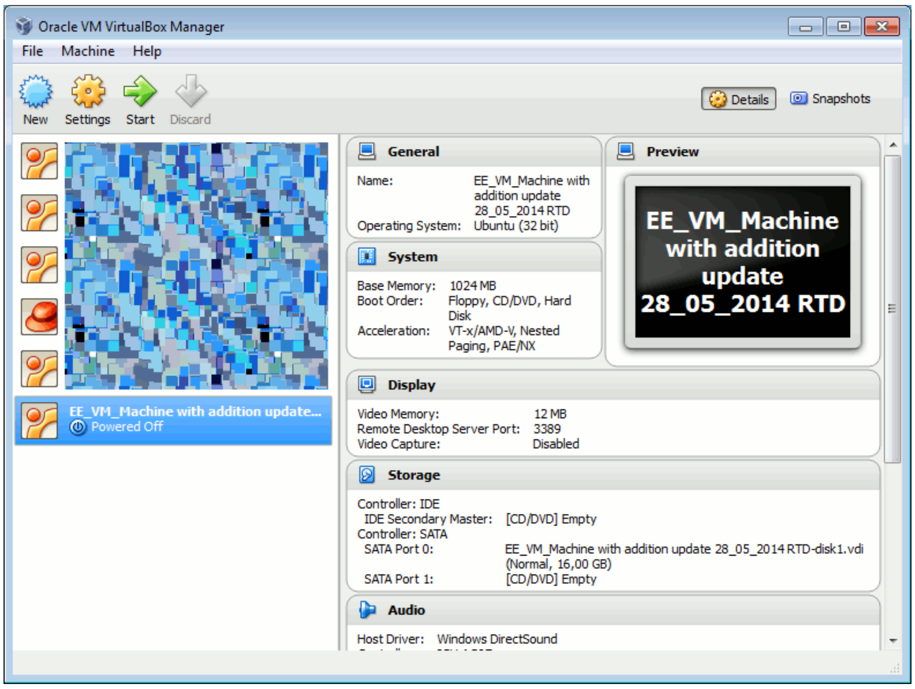
The Virtual Machine is typically distributed as a compressed file. Please decompress it. You will find at least two files, as in Figure 2.1. The file with the `vbox` extension is the file containing the settings of the virtual machine (describing the guest hardware, memory, disks, ...). The file with the `vdi` extension is the virtual hard disk used by the virtual machine.

On a typical VirtualBox setup, just double clicking on the file with `vbox` extension will open VirtualBox as in Figure 2.2. Just click on the Start button to boot the Virtual Machine.



Nome	Ultima modifica	Tipo	Dimensione
EE_VM_Machine with addition update 28_05_2014 RTD	12/06/2014 12:05	VirtualBox Machin...	10 KB
EE_VM_Machine with addition update 28_05_2014 RTD-disk1	12/06/2014 10:36	Virtual Disk Image	7.629.892 KB

Figure 2.1: Files obtained when unpacking the virtual machine. Actual file names may vary.


Oracle VM VirtualBox Manager
File
Machine
Help
New
Settings
Start
Discard
Details
Snapshots
EE_VM_Machine with addition update...
Powered Off
EE_VM_Machine with addition update 28_05_2014 RTD

General
Name: EE_VM_Machine with addition update 28_05_2014 RTD
Operating System: Ubuntu (32 bit)

System
Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Preview

Display
Video Memory: 12 MB
Remote Desktop Server Port: 3389
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [CD/DVD] Empty
Controller: SATA
SATA Port 0: EE_VM_Machine with addition update 28_05_2014 RTD-disk1.vdi (Normal, 16,00 GB)
SATA Port 1: [CD/DVD] Empty

Audio
Host Driver: Windows DirectSound

Figure 2.2: VirtualBox opened after clicking on the vbox file (the one with the blue icon).

2.3 VirtualBox settings

The following is a list of the main settings of the Virtual machine, useful if, for some reason, you need to recreate the `vbox` file from scratch. Those information must be set on a new virtual machine by clicking on the “Settings” button in Figure 2.2.

1. `General` Tab, `Basic` subtab: The Type of virtual machine must be Linux / Ubuntu 32 bit.

2. `System` tab: We suggest a 1Gb system memory, I/O APIC active, and as many processors as you have in your physical machine.

Please note that the virtual machine comes with the VirtualBox Guest Additions already installed. This turns out to be very convenient as the X Server will automatically recognize a resize of the VirtualBox window.

2.4 Starting the virtual machine

Once powered up, Linux will be boot, and the login screen of the Ubuntu Distribution will appear as in Figure 2.3.

To log in, use the following credentials:

Username: `evidence`

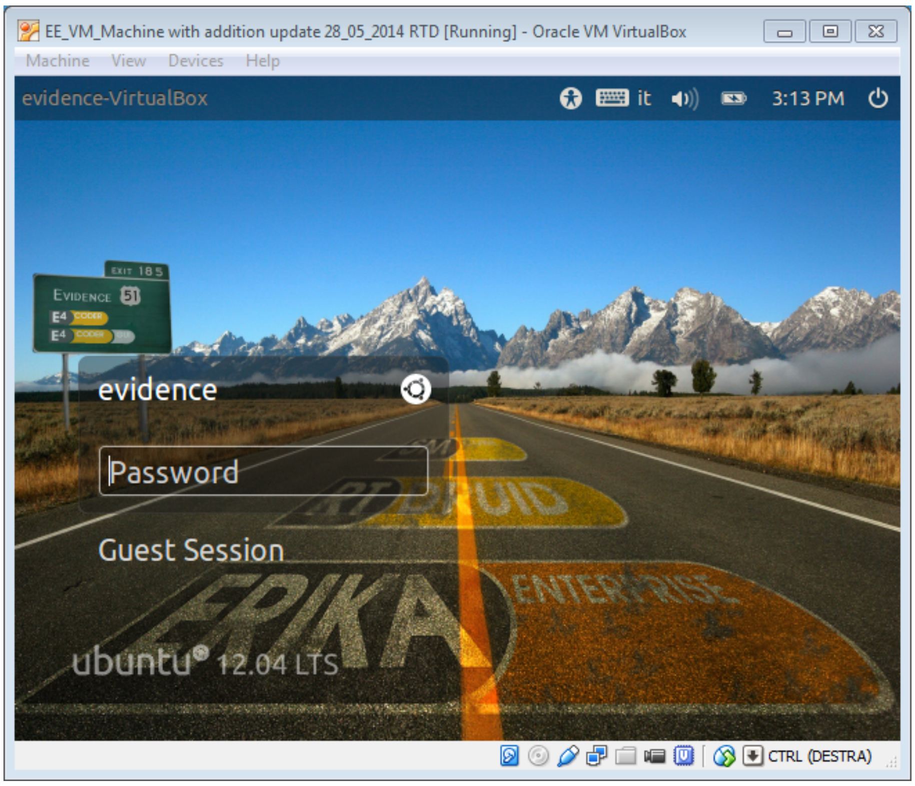


Figure 2.3: Login screen of the Virtual machine.

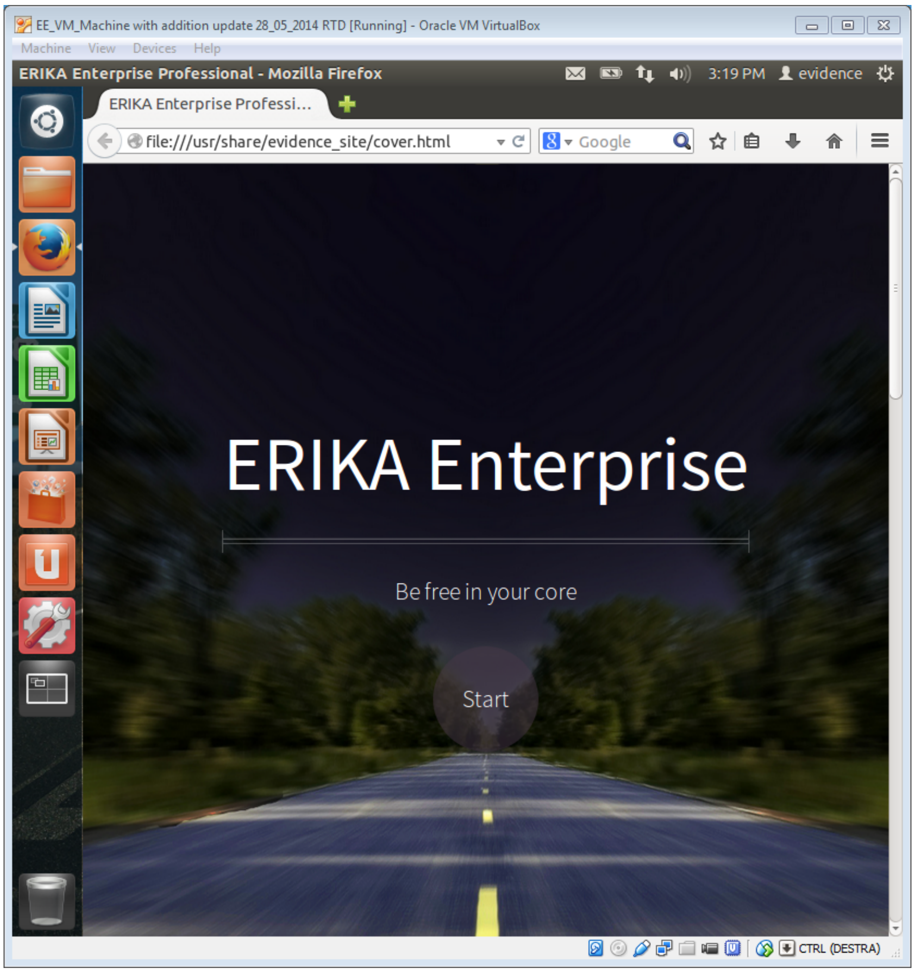

EE_VM_Machine with addition update 28_05_2014 RTD [Running] - Oracle VM VirtualBox
Machine View Devices Help
ERIKA Enterprise Professional - Mozilla Firefox
3:19 PM evidence
ERIKA Enterprise Professi...
file:///usr/share/evidence_site/cover.html
Google
ERIKA Enterprise
Be free in your core
Start
CTRL (DESTRA)

Figure 2.4: Welcome page.

Password: `evidence`

Once logged in, you will get an HTML welcome page as in Figure 2.4. The welcome page contains a set of documents on the following topics:

- Tutorial documentation (this file);
- Evidence Professional support on ERIKA Enterprise;
- Lauterbach TRACE32 additional information.

Finally, remember you can always type `RightCtrl-F` to go full screen.

3 Compiling the ERIKA Enterprise demo application

The following steps will guide you in the compilation of a simple ERIKA Enterprise application for the Freescale PPC MPC5674F Mamba:

1. To compile your first application with ERIKA Enterprise, you need to open the Eclipse IDE. There is an `Eclipse RT-Druid` link on the Desktop (see Figure 3.1).

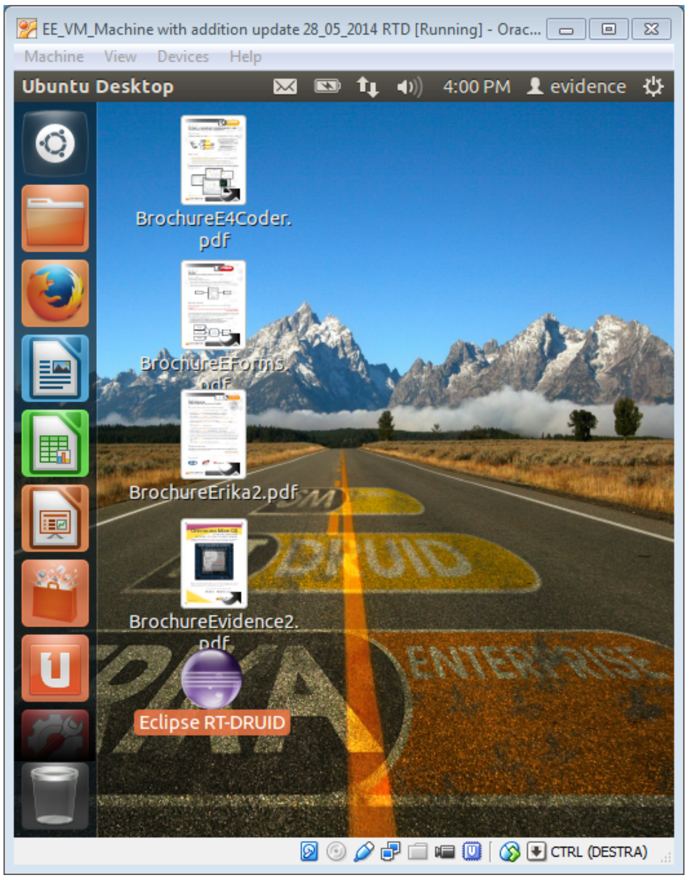
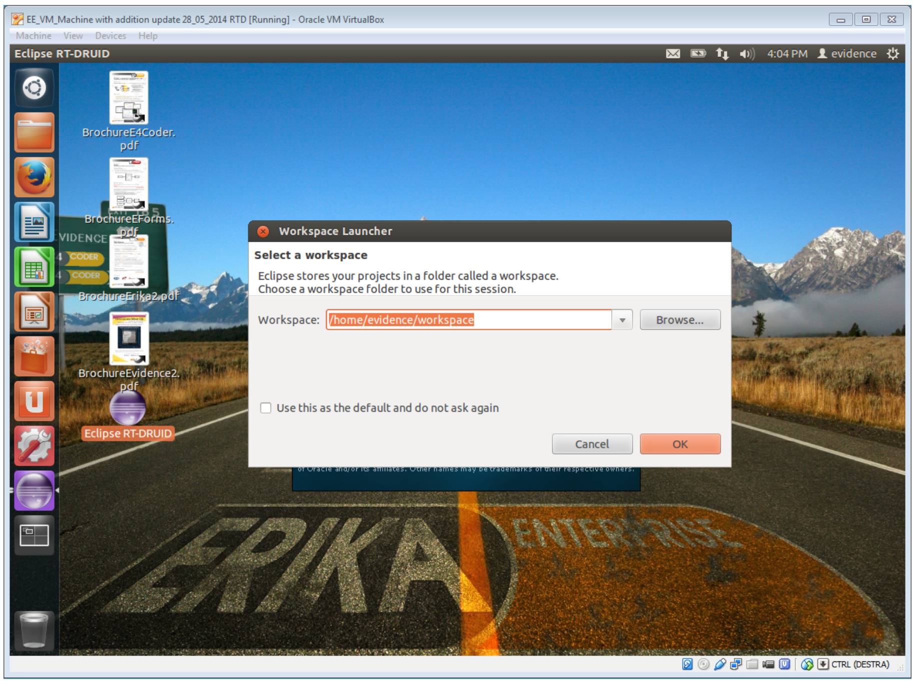


Figure 3.1: Please double click on the Eclipse icon to open RT-Druid.

2. Double click on it, and Eclipse will open requiring the workspace location.Please confirm the default location `/home/evidence/workspace` as in Figure 3.2. The Eclipse welcome screen will appear as in Figure 3.3. Click on the `Workbench` icon, and the default Eclipse view will appear.



Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/evidence/workspace

Use this as the default and do not ask again

Figure 3.2: Eclipse requires the workspace location. Leave the default setting.

3. Click on the "New" button in the toolbar (the first on the left), and choose "RT-Druid Oil and C/C++ project", as shown in Figure 3.4. A Dialog Box will appear.

4. Provide a name for the project. In our case, "Mamba_demo", as in Figure 3.5, and press Next.

5. Select the checkbox "Create a project using one of these templates", and select the `demo_sim` example as shown in Figure 3.6.

6. Click on the Finish button to create the example, as in Figure 3.7.

7. At this point, you can explore the demo example:

   - It is composed by two files, a conf.oil file, containing the OIL description needed to statically configure the kernel, and code.c, containing the application code.
   - The target board selected for this demo is an Axiom MPC5674evbfxmb evaluation board[1] hosting a Freescale MPC5674F Mamba microcontroller.
   - The demo is derived from the traditional *task* demo available for many architectures, and is composed by a high priority task blinking the first six LEDs in sequence, plus an additional task activated by a button, which changes the

[1]See http://erika.tuxfamily.org/wiki/index.php?title=Freescale_PPC_e200_(MPC_56xx) for more details about the ERIKA Enterprise support for PowerPC and for this board.

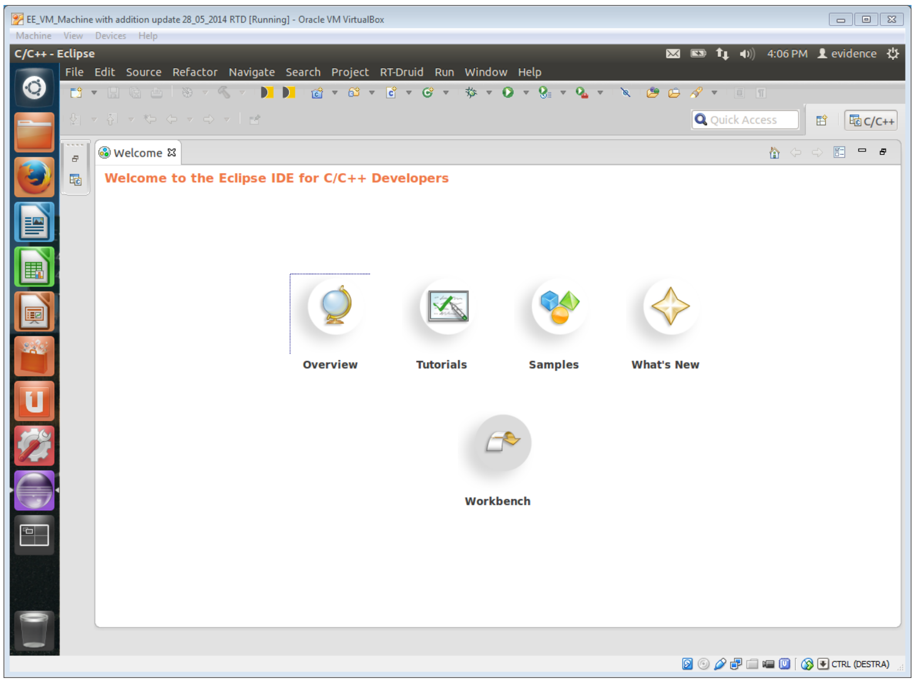

EE_VM_Machine with addition update 28_05_2014 RTD [Running] - Oracle VM VirtualBox
C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project RT-Druid Run Window Help
Quick Access
C/C++
Welcome
Welcome to the Eclipse IDE for C/C++ Developers
Overview
Tutorials
Samples
What's New
Workbench

Figure 3.3: The Eclipse welcome screen.

status of the other three LEDs. Other additional tasks are present as well doing minimal operations[2].

- The important settings in the OIL file are the following:

  `EE_OPT = ''__E200ZX_EXECUTE_FROM_RAM__'';`

  which sets the execution of the demo from RAM;

  `COMPILER_TYPE = GNU;`

  which selects the GCC crosscompiler for PPC which has been preinstalled on the virtual machine;

  `EE_OPT = ''EE_LAUTERBACH_DEMO_SIM'';`

  which copied additional Lauterbach TRACE32 scripts in the `Debug` directory.

8. To compile the project, just right click on the project name and select "Build Project" as shown in Figure 3.8. As a result, the project is compiled using the GCC compiler. The output is printed on the console view as in Figure 3.9 (you can discard the warnings in the "Problem" window as they are obtained by parsing the makefile output).

You are now ready to debug the compiled demo application using the Lauterbach TRACE32 debugger/tracer.

[2]Note that the original *task* demo had a different structure with various options to highlight the effects of preemptions and multiple activations.

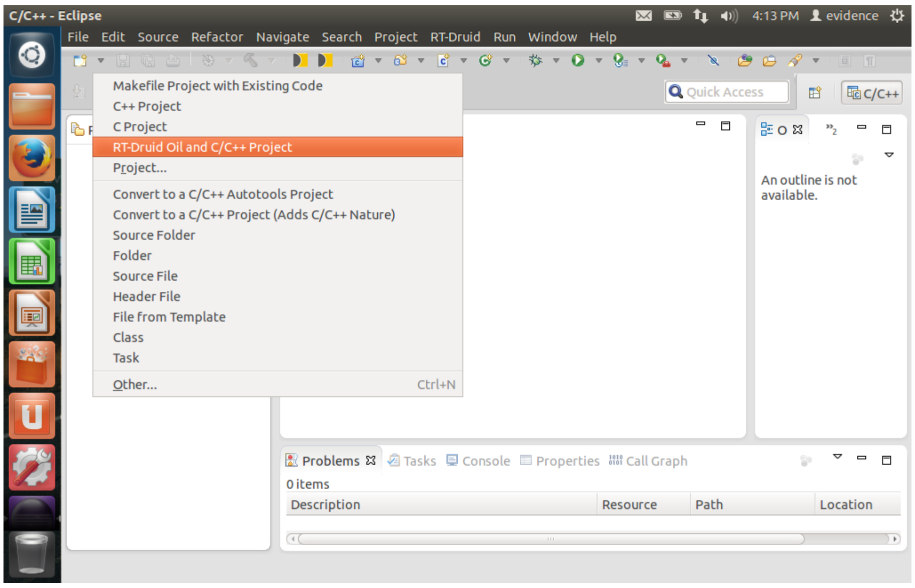
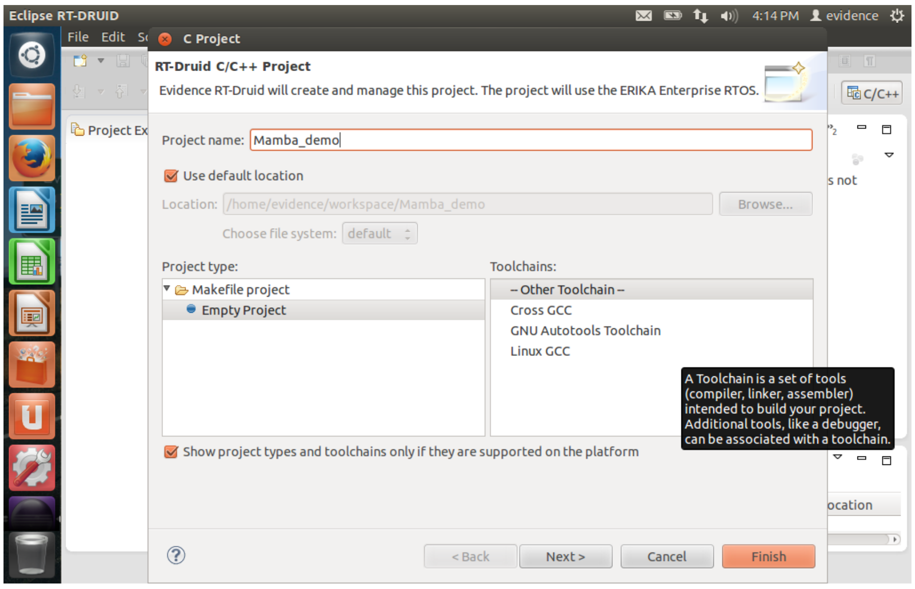

C/C++ - Eclipse
4:13 PM evidence
File Edit Source Refactor Navigate Search Project RT-Druid Run Window Help
Makefile Project with Existing Code
C++ Project
C Project
RT-Druid Oil and C/C++ Project
Project...
Convert to a C/C++ Autotools Project
Convert to a C/C++ Project (Adds C/C++ Nature)
Source Folder
Folder
Source File
Header File
File from Template
Class
Task
Other... Ctrl+N
Quick Access
C/C++
An outline is not available.
Problems
Tasks
Console
Properties
Call Graph
0 items
Description
Resource
Path
Location

Figure 3.4: Select “RT-Druid Oil and C/C++ Project” to create an RT-Druid project that will contain the demo.



RT-Druid C/C++ Project

Evidence RT-Druid will create and manage this project. The project will use the ERIKA Enterprise RTOS.

Project name: Mamba_demo

☑ Use default location

Location: /home/evidence/workspace/Mamba_demo Browse...

Choose file system: default

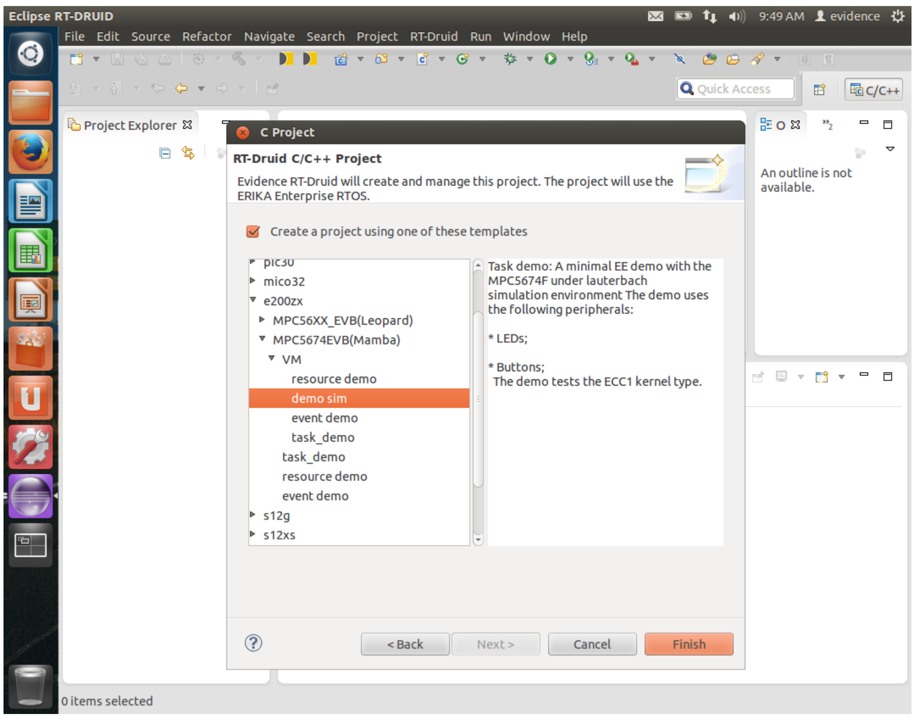
Project type:
- Makefile project
  - Empty Project

Toolchains:
- – Other Toolchain –
- Cross GCC
- GNU Autotools Toolchain
- Linux GCC

A Toolchain is a set of tools (compiler, linker, assembler) intended to build your project. Additional tools, like a debugger, can be associated with a toolchain.

☑ Show project types and toolchains only if they are supported on the platform

Figure 3.5: Provide a name for the Eclipse project containing the demo.



RT-Druid C/C++ Project

Evidence RT-Druid will create and manage this project. The project will use the ERIKA Enterprise RTOS.

Create a project using one of these templates

```
▸ pic30
▸ mico32
▾ e200zx
  ▸ MPC56XX_EVB(Leopard)
  ▾ MPC5674EVB(Mamba)
    ▾ VM
        resource demo
        demo sim
        event demo
        task_demo
      task_demo
      resource demo
      event demo
▸ s12g
▸ s12xs
```

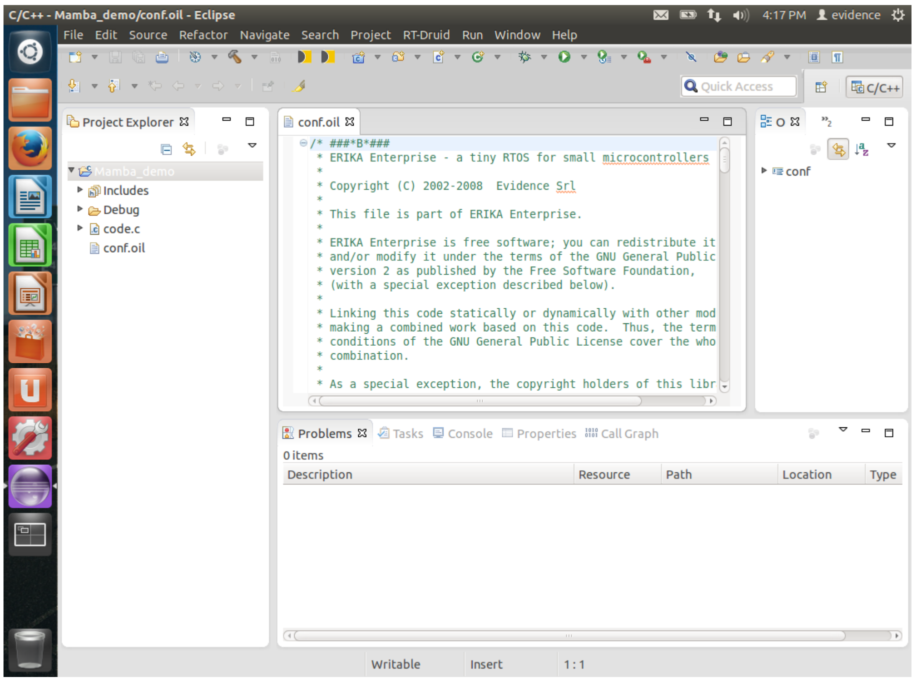
Task demo: A minimal EE demo with the MPC5674F under lauterbach simulation environment The demo uses the following peripherals:

* LEDs;

* Buttons;
The demo tests the ECC1 kernel type.

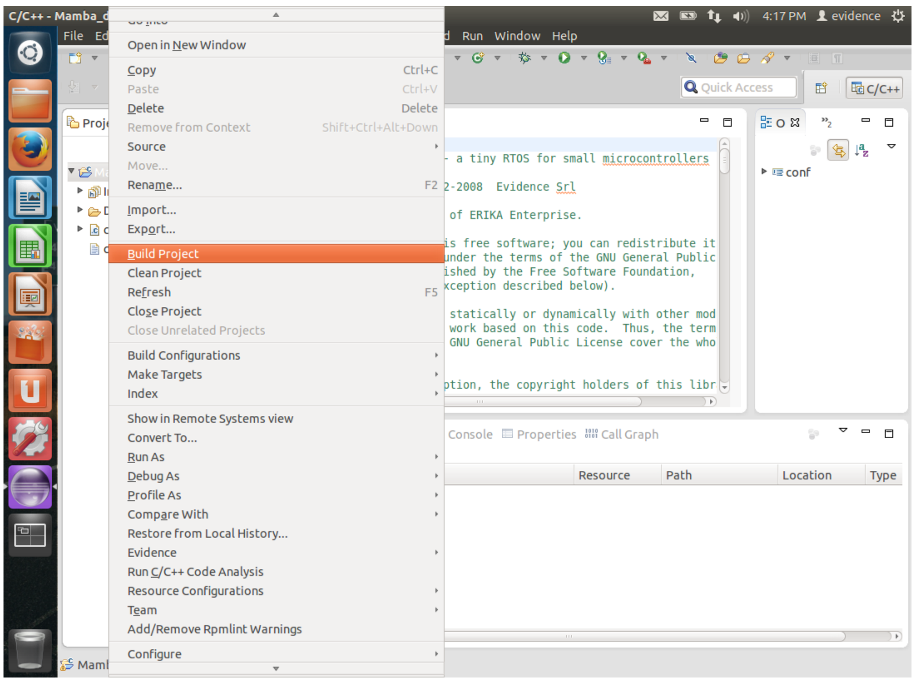
An outline is not available.

Figure 3.6: Selecting the demo code for the template.



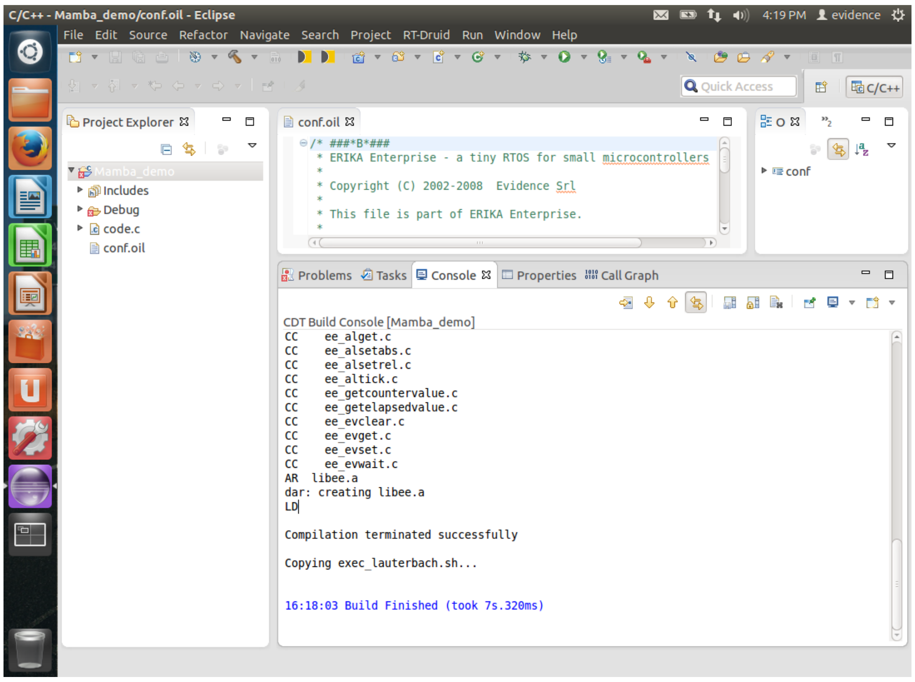
```
/* ###*B*###
 * ERIKA Enterprise - a tiny RTOS for small microcontrollers
 *
 * Copyright (C) 2002-2008  Evidence Srl
 *
 * This file is part of ERIKA Enterprise.
 *
 * ERIKA Enterprise is free software; you can redistribute it
 * and/or modify it under the terms of the GNU General Public
 * version 2 as published by the Free Software Foundation,
 * (with a special exception described below).
 *
 * Linking this code statically or dynamically with other mod
 * making a combined work based on this code.  Thus, the term
 * conditions of the GNU General Public License cover the who
 * combination.
 *
 * As a special exception, the copyright holders of this libr
```

Figure 3.7: Eclipse opened with the demo example.



```
- a tiny RTOS for small microcontrollers
2-2008  Evidence Srl
 of ERIKA Enterprise.
is free software; you can redistribute it
under the terms of the GNU General Public
ished by the Free Software Foundation,
xception described below).
 statically or dynamically with other mod
 work based on this code.  Thus, the term
 GNU General Public License cover the who
ption, the copyright holders of this libr
```

Figure 3.8: Select “Build project” to compile the demo example.



```
/* ###*B*###
 * ERIKA Enterprise - a tiny RTOS for small microcontrollers
 *
 * Copyright (C) 2002-2008  Evidence Srl
 *
 * This file is part of ERIKA Enterprise.
 *
```

```
CDT Build Console [Mamba_demo]
CC    ee_alget.c
CC    ee_alsetabs.c
CC    ee_alsetrel.c
CC    ee_altick.c
CC    ee_getcountervalue.c
CC    ee_getelapsedvalue.c
CC    ee_evclear.c
CC    ee_evget.c
CC    ee_evset.c
CC    ee_evwait.c
AR  libee.a
dar: creating libee.a
LD

Compilation terminated successfully

Copying exec_lauterbach.sh...


16:18:03 Build Finished (took 7s.320ms)
```

Figure 3.9: The build results are displayed in the Console View.

4 Debugging the demo using Lauterbach TRACE32

4.1 Starting TRACE32

Once compiled, the demo is ready to be programmed on the target board. In our case we will not directly program the board, but we will execute the binary generated by the demo compilation using the Lauterbach TRACE32 Instruction Set Simulator (ISS).

An Instruction Set Simulator is basically a program able to "emulate" the execution of the code as it was executed on the real target microcontroller. In addition to this, Lauterbach provides for this demo an additional emulation for 8 LEDs and one button, capable of generating interrupts. In this way, it is possible to directly interact with the simulator as you would normally do with the demo running on the real evaluation board.

To run the executable generated during the compilation with the TRACE32 ISS, you need to follow these simple steps:

1. Open a Terminal window as shown in Figure 4.1.
2. Change directory into the Debug directory created by RT-Druid when compiling the demo (see Figure 4.2).
3. Launch the script

   ```
   ./exec_lauterbach
   ```

   to launch TRACE32. A window like the one in Figure 4.3 will appear.

4.2 How to use TRACE32 with the demo

The `./exec_lauterbach` script is basically a wrapper over the TRACE32 executable `t32mppc-qt`. TRACE32 at boot will load the following startup scripts:

- `t32.cmm` is the startup script that is automatically executed at startup. It reprograms the TRACE32 GUI adding further buttons to the toolbar, and then it calls another script named `demo.cmm`;
- `demo.cmm` performs the following actions:
  - it prepares the CPU setup in TRACE32 instruction set simulator;
  - it loads the ERIKA binary object and symbol information;

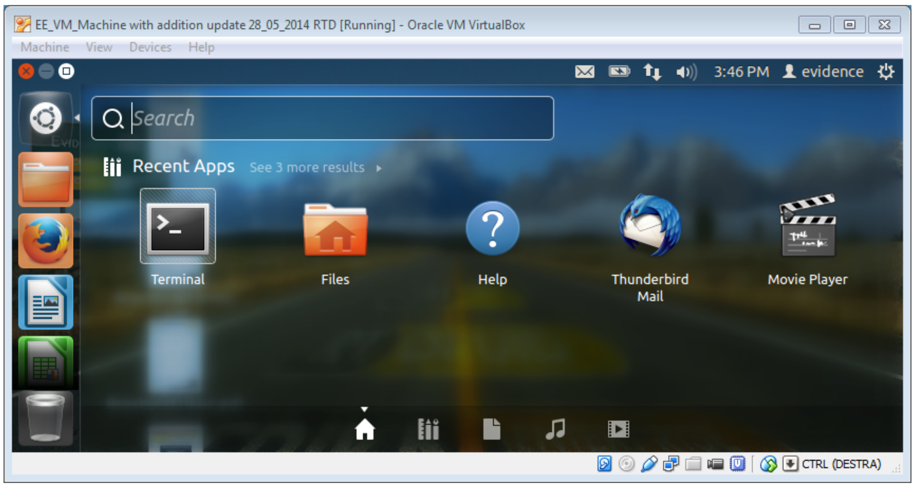
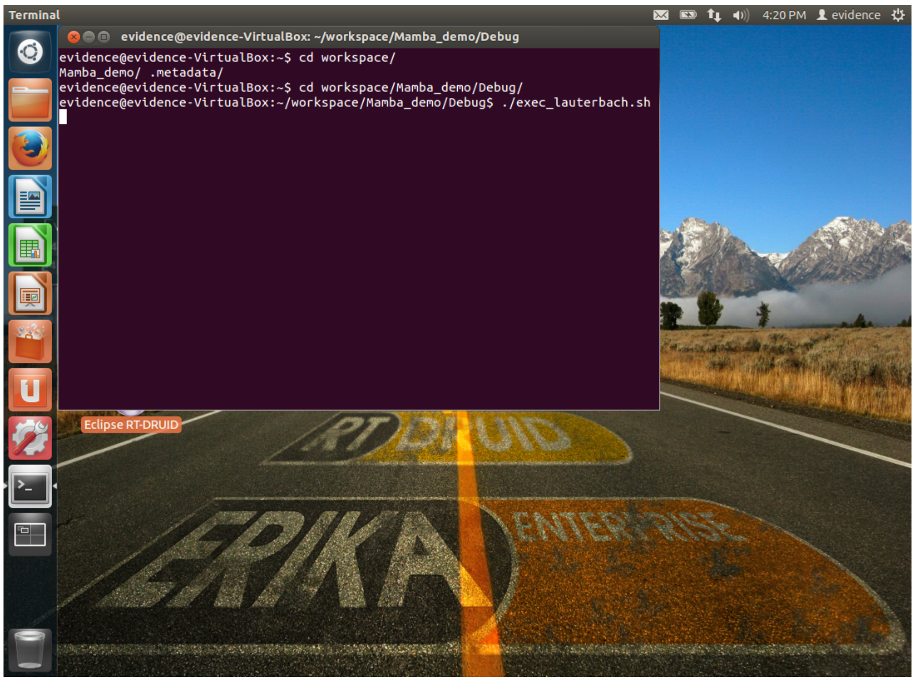
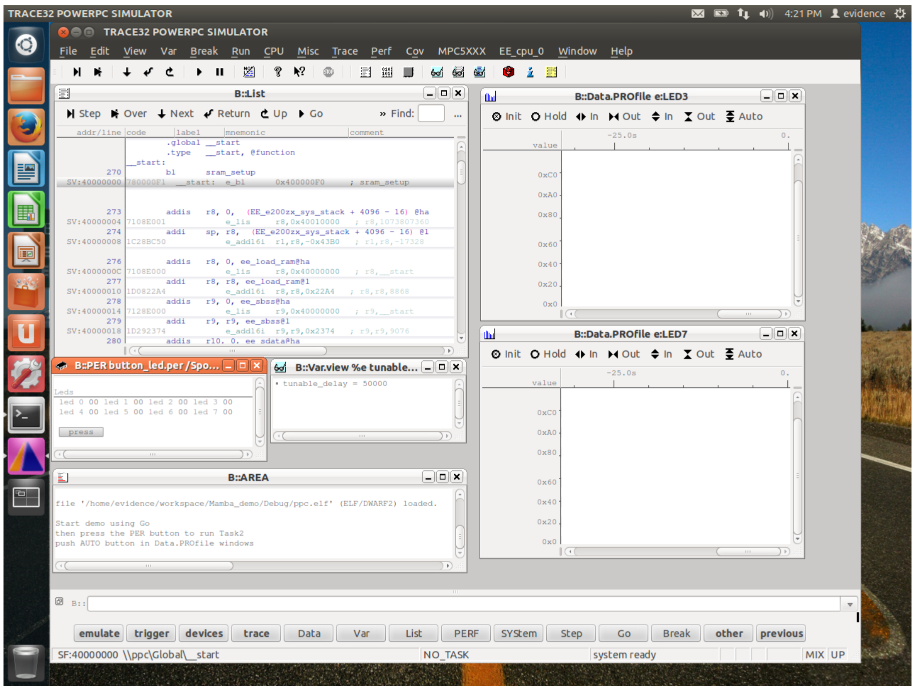


Figure 4.1: Open a terminal.



```
evidence@evidence-VirtualBox:~$ cd workspace/
Mamba_demo/ .metadata/
evidence@evidence-VirtualBox:~$ cd workspace/Mamba_demo/Debug/
evidence@evidence-VirtualBox:~/workspace/Mamba_demo/Debug$ ./exec_lauterbach.sh
```

Figure 4.2: Change directory and run the `exec_lauterbach` script to launch TRACE32.


TRACE32 POWERPC SIMULATOR
File Edit View Var Break Run CPU Misc Trace Perf Cov MPC5XXX EE_cpu_0 Window Help
B::List
Step Over Next Return Up Go Find:
B::Data.PROfile e:LED3
Init Hold In Out In Out Auto
B::Data.PROfile e:LED7
B::PER button_led.per /Spo...
B::Var.view %e tunable...
B::AREA
emulate trigger devices trace Data Var List PERF SYStem Step Go Break other previous
SF:40000000 \\ppc\Global\__start
NO_TASK
system ready
MIX UP

```
addr/line  code       label     mnemonic                                    comment
                               .global __start
                               .type   __start, @function
                     __start:
      270                      bl      sram_setup
SV:40000000 780000F1  __start:  e_bl      0x400000F0      ; sram_setup

      273                      addis   r8, 0,  (EE_e200zx_sys_stack + 4096 - 16) @ha
SV:40000004 7108E001            e_lis     r8,0x40010000   ; r8,1073807360
      274                      addi    sp, r8,  (EE_e200zx_sys_stack + 4096 - 16) @l
SV:40000008 1C28BC50            e_add16i  r1,r8,-0x43B0   ; r1,r8,-17328

      276                      addis   r8, 0, ee_load_ram@ha
SV:4000000C 7108E000            e_lis     r8,0x40000000   ; r8,__start
      277                      addi    r8, r8, ee_load_ram@l
SV:40000010 1D0822A4            e_add16i  r8,r8,0x22A4    ; r8,r8,8868
      278                      addis   r9, 0, ee_sbss@ha
SV:40000014 7128E000            e_lis     r9,0x40000000   ; r9,__start
      279                      addi    r9, r9, ee_sbss@l
SV:40000018 1D292374            e_add16i  r9,r9,0x2374    ; r9,r9,9076
      280                      addis   r10, 0, ee_sdata@ha
```

```
Leds
 led 0 00 led 1 00 led 2 00 led 3 00
 led 4 00 led 5 00 led 6 00 led 7 00

 press
```

```
• tunable_delay = 50000
```

```
file '/home/evidence/workspace/Mamba_demo/Debug/ppc.elf' (ELF/DWARF2) loaded.

Start demo using Go
then press the PER button to run Task2
push AUTO button in Data.PROfile windows
```

Figure 4.3: Lauterbach TRACE32 configured to run the demo.

- it loads the TRACE32 OSEK/ORTI awareness;
- it sets up some windows in the GUI.

The windows page system in the TRACE32 GUI can be controlled with mouse right-click on the toolbar. A set of three selectable pages will be available as shown in Figure 4.4. The available pages are described below:

- `Debug`: shows the basic debug controls in the List window. Debug controls (Go / Break / Step / . . .) are also available in the main GUI toolbar. A `PER` (PERipheral) window is present to simulate the periodic toggle of six leds, done by the ERIKA task `Task1`. The "press" button in the `PER` window, allows to alternate high /

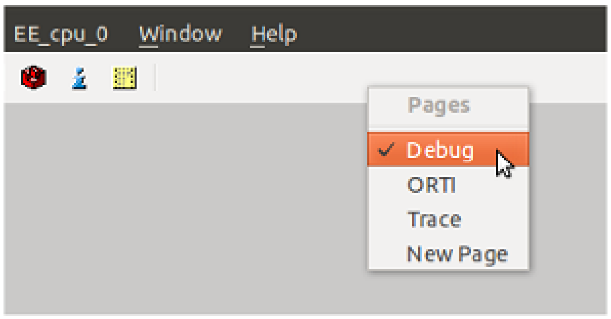
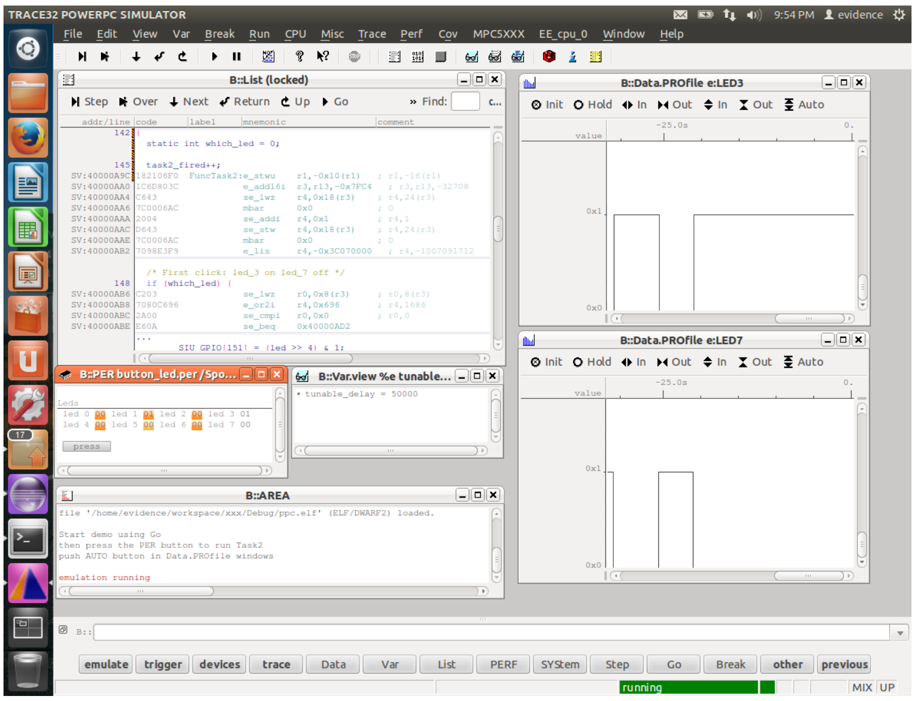


Figure 4.4: The Lauterbach TRACE32 pages available with right-click.


TRACE32 POWERPC SIMULATOR
File Edit View Var Break Run CPU Misc Trace Perf Cov MPC5XXX EE_cpu_0 Window Help
B::List (locked)
Step Over Next Return Up Go Find:
B::Data.PROfile e:LED3
Init Hold In Out In Out Auto
B::Data.PROfile e:LED7
B::PER button_led.per /Spo...
B::Var.view %e tunable...
B::AREA
press
emulate trigger devices trace Data Var List PERF SYStem Step Go Break other previous
running
MIX UP

```
addr/line   code      label      mnemonic                comment
      142 {
             static int which_led = 0;

      145    task2_fired++;
SV:40000A9C 182106F0  FuncTask2:e_stwu   r1,-0x10(r1)      ; r1,-16(r1)
SV:40000AA0 1C6D803C            e_add16i r3,r13,-0x7FC4    ; r3,r13,-32708
SV:40000AA4 C643                se_lwz   r4,0x18(r3)       ; r4,24(r3)
SV:40000AA6 7C0006AC            mbar     0x0               ; 0
SV:40000AAA 2004                se_addi  r4,0x1            ; r4,1
SV:40000AAC D643                se_stw   r4,0x18(r3)       ; r4,24(r3)
SV:40000AAE 7C0006AC            mbar     0x0               ; 0
SV:40000AB2 7098E3F9            e_lis    r4,-0x3C070000    ; r4,-1007091712

             /* First click: led_3 on led_7 off */
      148    if (which_led) {
SV:40000AB6 C203                se_lwz   r0,0x8(r3)        ; r0,8(r3)
SV:40000AB8 7080C696            e_or2i   r4,0x696          ; r4,1686
SV:40000ABC 2A00                se_cmpi  r0,0x0            ; r0,0
SV:40000ABE E60A                se_beq   0x40000AD2
...
                SIU_GPIO[151] = (led >> 4) & 1;
```

```
Leds
 led 0 00 led 1 01 led 2 00 led 3 01
 led 4 00 led 5 00 led 6 00 led 7 00
```

```
• tunable_delay = 50000
```

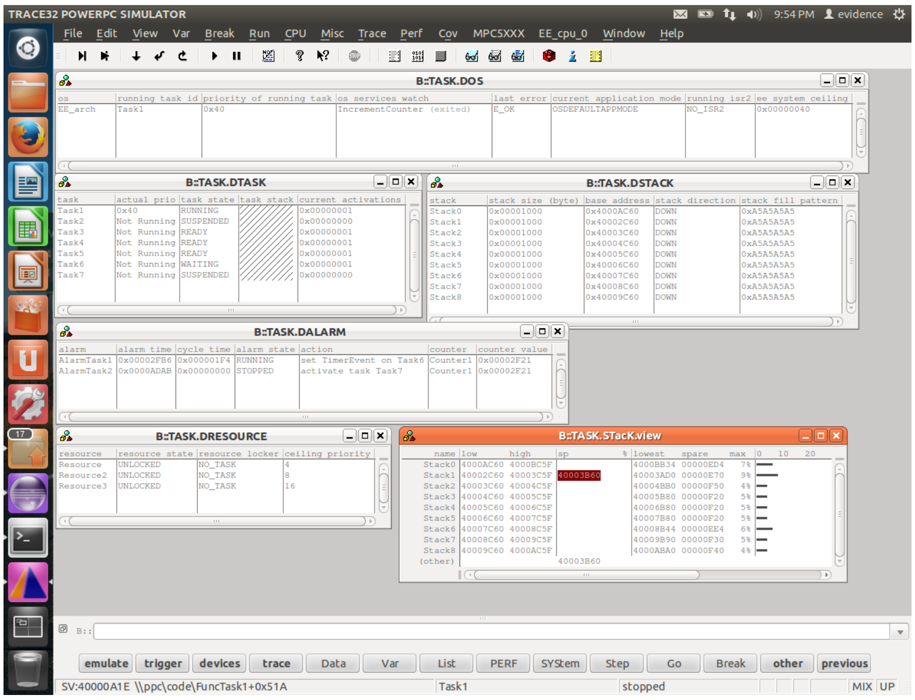
```
file '/home/evidence/workspace/xxx/Debug/ppc.elf' (ELF/DWARF2) loaded.

Start demo using Go
then press the PER button to run Task2
push AUTO button in Data.PROfile windows

emulation running
```

Figure 4.5: The Debug window of Lauterbach TRACE32. Please note the PER window, and the window showing the LED status sampled at runtime.

low values at two other simulated leds. This is done calling another ERIKA task `Task2`. A couple of `Data.PROfile` windows are present for profiling of the values of these two leds. Press the “Auto” button to show the values sampled over time. The `Var.view` window let the user modify a program variable, which controls the led blinking delay, for a fine tuning. A screenshot of this window is available in Figure 4.5.

- `ORTI`: shows the basic OSEK / ORTI views provided by the TRACE32 RTOS awareness, for real time, non-intrusive display of OSEK system resources, including stack space usage. A screenshot of this window is available in Figure 4.6.
- `Trace`: shows some trace-related features available in TRACE32. Both statistics and graphics views are available with trace information at symbol or task level. In the graphics views, the user can enlarge or reduce the time scale, using the mouse wheel. Some windows are “tracked” each other, so that clicking with the mouse in a trace window changes the tracked windows accordingly. The `Trace.List` window shows the recorded program and data trace. “More” and “Less” buttons are present to show or suppress details of trace information, from HLL view to assembly code and data access cycles. A screenshot of this window is available in Figure 4.7.

More functions are available in the TRACE32 menus and in the toolbar. For more information please refer to `http://www.lauterbach.com/pdf/training_hll.pdf`.


TRACE32 POWERPC SIMULATOR
9:54 PM
evidence
File Edit View Var Break Run CPU Misc Trace Perf Cov MPC5XXX EE_cpu_0 Window Help
B::
emulate trigger devices trace Data Var List PERF SYStem Step Go Break other previous
SV:40000A1E \\ppc\code\FuncTask1+0x51A
Task1
stopped
MIX UP

B::TASK.DOS

os	running task id	priority of running task	os services watch	last error	current application mode	running isr2	ee system ceiling
EE_arch	Task1	0x40	IncrementCounter (exited)	E_OK	OSDEFAULTAPPMODE	NO_ISR2	0x00000040

B::TASK.DTASK

task	actual prio	task state	task stack	current activations
Task1	0x40	RUNNING	▨	0x00000001
Task2	Not Running	SUSPENDED	▨	0x00000000
Task3	Not Running	READY	▨	0x00000001
Task4	Not Running	READY	▨	0x00000001
Task5	Not Running	READY	▨	0x00000001
Task6	Not Running	WAITING	▨	0x00000001
Task7	Not Running	SUSPENDED	▨	0x00000000

B::TASK.DSTACK

stack	stack size (byte)	base address	stack direction	stack fill pattern
Stack0	0x00001000	0x4000AC60	DOWN	0xA5A5A5A5
Stack1	0x00001000	0x40002C60	DOWN	0xA5A5A5A5
Stack2	0x00001000	0x40003C60	DOWN	0xA5A5A5A5
Stack3	0x00001000	0x40004C60	DOWN	0xA5A5A5A5
Stack4	0x00001000	0x40005C60	DOWN	0xA5A5A5A5
Stack5	0x00001000	0x40006C60	DOWN	0xA5A5A5A5
Stack6	0x00001000	0x40007C60	DOWN	0xA5A5A5A5
Stack7	0x00001000	0x40008C60	DOWN	0xA5A5A5A5
Stack8	0x00001000	0x40009C60	DOWN	0xA5A5A5A5

B::TASK.DALARM

alarm	alarm time	cycle time	alarm state	action	counter	counter value
AlarmTask1	0x00002FB6	0x000001F4	RUNNING	set TimerEvent on Task6	Counter1	0x00002F21
AlarmTask2	0x0000ADAB	0x00000000	STOPPED	activate task Task7	Counter1	0x00002F21

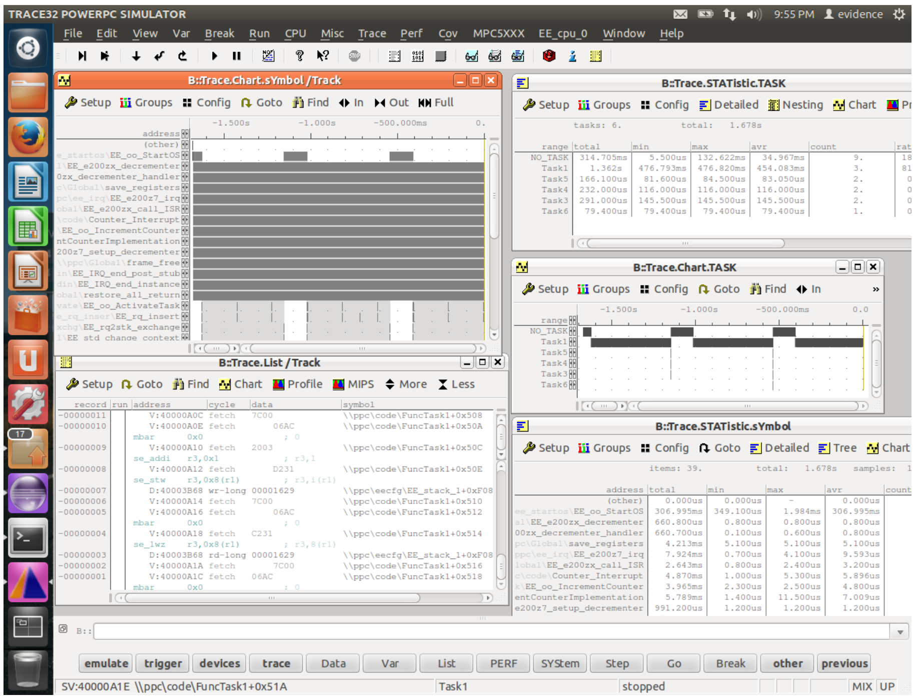
B::TASK.DRESOURCE

resource	resource state	resource locker	ceiling priority
Resource	UNLOCKED	NO_TASK	4
Resource2	UNLOCKED	NO_TASK	8
Resource3	UNLOCKED	NO_TASK	16

B::TASK.STacK.view

name	low	high	sp	%	lowest	spare	max	0 10 20
Stack0	4000AC60	4000BC5F			4000BB34	00000ED4	7%	▬
Stack1	40002C60	40003C5F	40003B60		40003AD0	00000E70	9%	▬
Stack2	40003C60	40004C5F			40004BB0	00000F50	4%	▬
Stack3	40004C60	40005C5F			40005B80	00000F20	5%	▬
Stack4	40005C60	40006C5F			40006B80	00000F20	5%	▬
Stack5	40006C60	40007C5F			40007B80	00000F20	5%	▬
Stack6	40007C60	40008C5F			40008B44	00000EE4	6%	▬
Stack7	40008C60	40009C5F			40009B90	00000F30	5%	▬
Stack8	40009C60	4000AC5F			4000ABA0	00000F40	4%	▬
(other)			40003B60					

Figure 4.6: The ORTI window of Lauterbach TRACE32. Please note the stack usage view derived from the fillpattern specified in the ORTI file.



B::Trace.STATistic.TASK

Setup Groups Config Detailed Nesting Chart Pr

tasks: 6. total: 1.678s

range	total	min	max	avr	count	rat
NO_TASK	314.705ms	5.500us	132.622ms	34.967ms	9.	18
Task1	1.362s	476.793ms	476.820ms	454.083ms	3.	81
Task5	166.100us	81.600us	84.500us	83.050us	2.	0
Task4	232.000us	116.000us	116.000us	116.000us	2.	0
Task3	291.000us	145.500us	145.500us	145.500us	2.	0
Task6	79.400us	79.400us	79.400us	79.400us	1.	0

B::Trace.List /Track

Setup Goto Find Chart Profile MIPS More Less

```
record   |run|address      cycle     data          symbol
-00000011      V:40000A0C fetch    7C00            \\ppc\code\FuncTask1+0x508
-00000010      V:40000A0E fetch        06AC        \\ppc\code\FuncTask1+0x50A
            mbar       0x0              ; 0
-00000009      V:40000A10 fetch    2003            \\ppc\code\FuncTask1+0x50C
            se_addi    r3,0x1           ; r3,1
-00000008      V:40000A12 fetch        D231        \\ppc\code\FuncTask1+0x50E
            se_stw     r3,0x8(r1)       ; r3,i(r1)
-00000007      D:40003B68 wr-long 00001629         \\ppc\eecfg\EE_stack_1+0xF08
-00000006      V:40000A14 fetch    7C00            \\ppc\code\FuncTask1+0x510
-00000005      V:40000A16 fetch        06AC        \\ppc\code\FuncTask1+0x512
            mbar       0x0              ; 0
-00000004      V:40000A18 fetch    C231            \\ppc\code\FuncTask1+0x514
            se_lwz     r3,0x8(r1)       ; r3,8(r1)
-00000003      D:40003B68 rd-long 00001629         \\ppc\eecfg\EE_stack_1+0xF08
-00000002      V:40000A1A fetch        7C00        \\ppc\code\FuncTask1+0x516
-00000001      V:40000A1C fetch    06AC            \\ppc\code\FuncTask1+0x518
            mbar       0x0              ; 0
```

B::Trace.STATistic.sYmbol

Setup Groups Config Goto Detailed Tree Chart

items: 39. total: 1.678s samples: 1

address	total	min	max	avr	count
(other)	0.000us	0.000us	-	0.000us	
ee_startos\EE_oo_StartOS	306.995ms	349.100us	1.984ms	306.995ms	
al\EE_e200zx_decrementer	660.800us	0.800us	0.800us	0.800us	
00zx_decrementer_handler	660.700us	0.100us	0.600us	0.800us	
pc\Global\save_registers	4.213ms	5.100us	5.100us	5.100us	
ppc\ee_irq\EE_e200z7_irq	7.924ms	0.700us	4.100us	9.593us	
lobal\EE_e200zx_call_ISR	2.643ms	0.800us	2.400us	3.200us	
c\code\Counter_Interrupt	4.870ms	1.000us	5.300us	5.896us	
k\EE_oo_IncrementCounter	3.965ms	2.300us	2.500us	4.800us	
entCounterImplementation	5.789ms	1.400us	11.500us	7.009us	
e200z7_setup_decrementer	991.200us	1.200us	1.200us	1.200us	

Figure 4.7: The Trace window of Lauterbach TRACE32. Please note context switch Gantt chart, and the timing measurements for the application tasks.

5 Acknowledgments

We would like to acknowledge Maurizio Menegotto, Marco Ferrario, Rudolf Dienstbeck from Lauterbach, and Bruno Morelli, Francesco Esposito, Giuseppe Serano from Evidence for the support provided during the development of the demos, and during integration phase of this Virtual Machine.