ORTI Plug-in for the IAR Embedded Workbench C-SPY Debugger

Contents

Introduction

The ORTI plug-in introduces the following elements in the C-SPY user interface.

- On the **View** menu, a new entry, **ORTI**, is added from which a number of specific windows can be opened.
- A new menu item, **ORTI**, is added with entries for task-related stepping commands and options.
- A new toolbar with buttons corresponding to the task-related stepping commands on the **ORTI** menu is added.
- In the **Breakpoints dialog box** a new button, **Task…**, is added, making it possible for standard breakpoints to be task-specific.

The following criteria must be met for the ORTI plug-in to be enabled.

- The function label `main` must exist and be executed.

Optional ORTI plug-in functionality.

- To enable *task awareness* the ORTI description file must contain the `TASK` object type and the `OS` object type with the `RUNNINGTASK` attribute.
- To enable *service trace* the ORTI description file must contain the `OS` object type with the `SERVICETRACE` attribute.

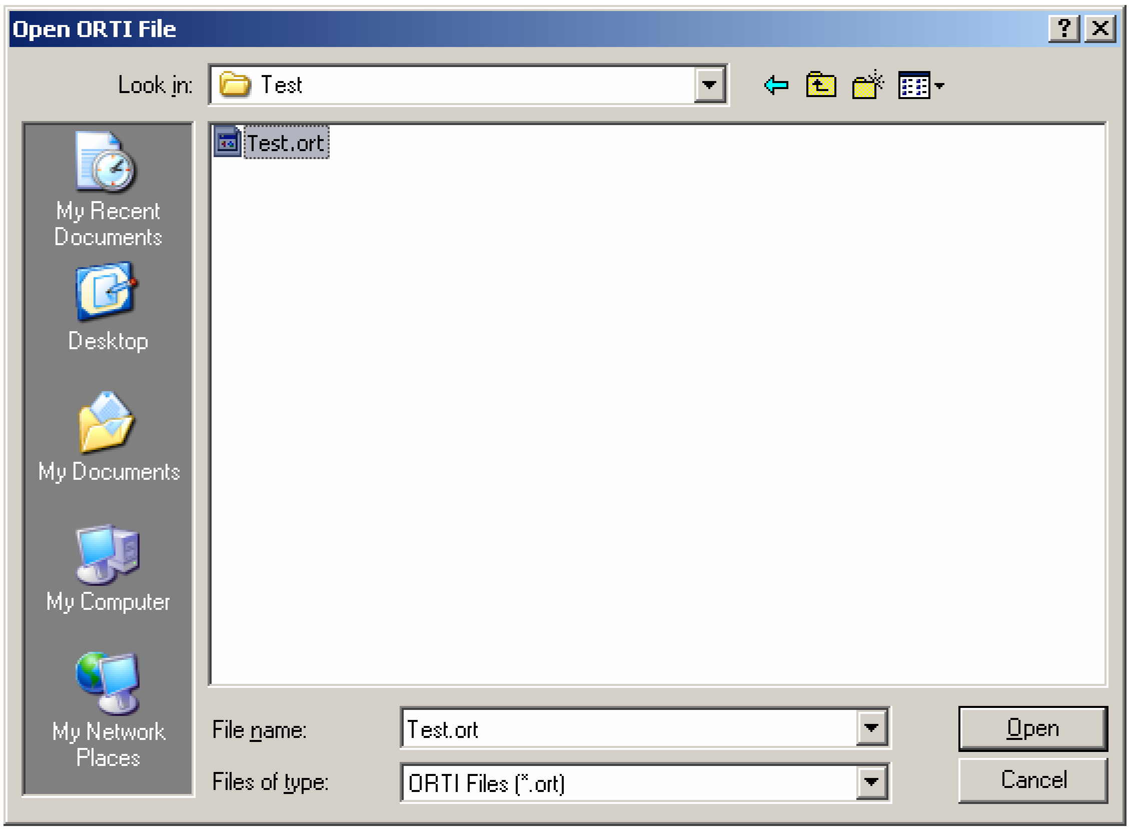
Setup

Standard plug-in setup. See the product *IAR Embedded Workbench IDE User Guide*.

ORTI input file

The ORTI plug-in needs a description file as input. How this file is generated is the responsibility of the RTOS vendor. The file must be compatible with the *OSEK Run Time Interface (ORTI)*, *Part A (Version 2.1.1)* and *Part B (Version 2.1)*.

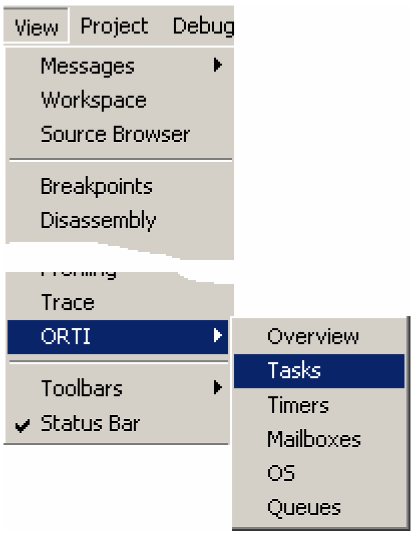
When the ORTI plug-in is activated for the first time for a project a file dialog box will pop-up allowing the user to input the ORTI description file to use:



The filename will be saved in the debug session settings file and need not to be entered again.

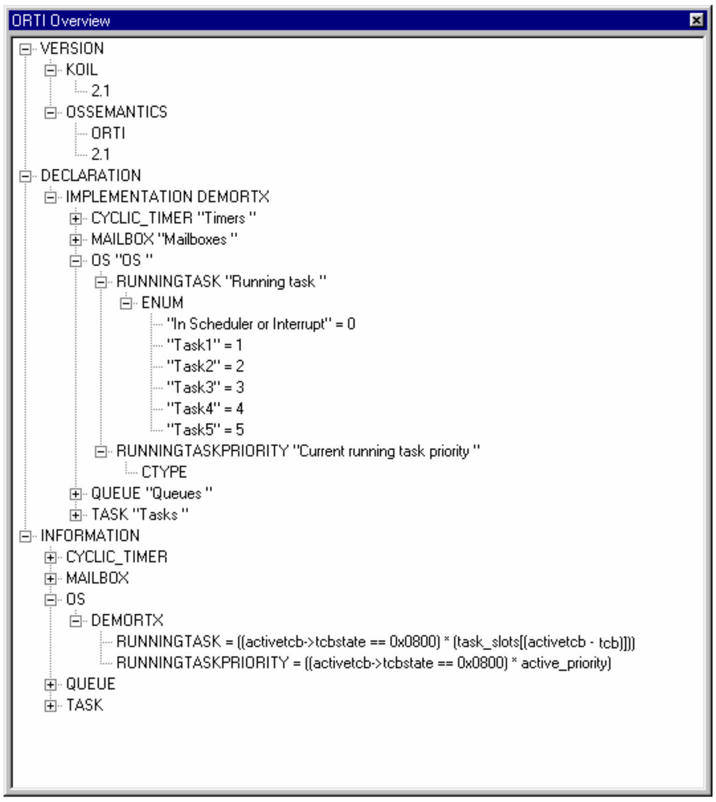
Windows

The ORTI plug-in introduces a number of debugger windows. Only the first, Overview, has a fixed name; the others depend on the information provided in the ORTI input file.
The windows can be found on the **View** menu, for example:



The Overview Window

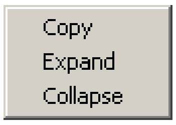
This window shows an overview of the data imported from the ORTI input file.



```
ORTI Overview
⊟ VERSION
   ⊟ KOIL
      2.1
   ⊟ OSSEMANTICS
      ORTI
      2.1
⊟ DECLARATION
   ⊟ IMPLEMENTATION DEMORTX
      ⊞ CYCLIC_TIMER "Timers "
      ⊞ MAILBOX "Mailboxes "
      ⊟ OS "OS "
         ⊟ RUNNINGTASK "Running task "
            ⊟ ENUM
               "In Scheduler or Interrupt" = 0
               "Task1" = 1
               "Task2" = 2
               "Task3" = 3
               "Task4" = 4
               "Task5" = 5
         ⊟ RUNNINGTASKPRIORITY "Current running task priority "
            CTYPE
      ⊞ QUEUE "Queues "
      ⊞ TASK "Tasks "
⊟ INFORMATION
   ⊞ CYCLIC_TIMER
   ⊞ MAILBOX
   ⊟ OS
      ⊟ DEMORTX
         RUNNINGTASK = ((activetcb->tcbstate == 0x0800) * (task_slots[(activetcb - tcb)]))
         RUNNINGTASKPRIORITY = ((activetcb->tcbstate == 0x0800) * active_priority)
   ⊞ QUEUE
   ⊞ TASK
```

Overview window context menu

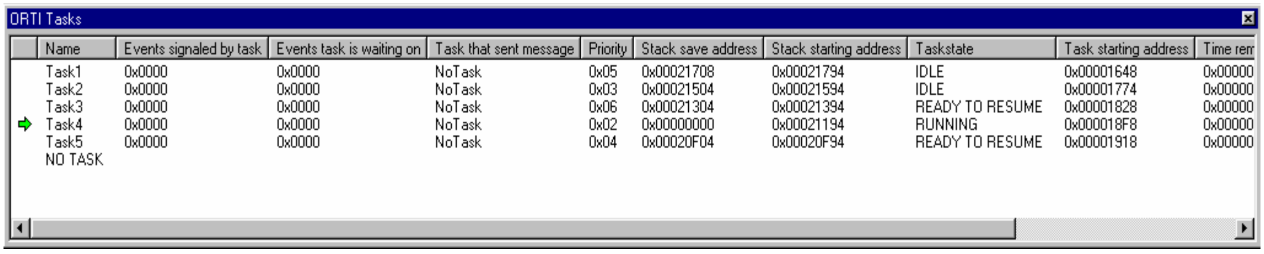
The context menu available in the Overview window provides commands for copy, expand, and collapse of the selected item.



Inspection Windows

Task Inspection Window

If the ORTI description file contains the `TASK` object type and the `OS` object type with the `RUNNINGTASK` attribute, a task aware task inspection window is available. The name of the window is vendor specific but the window, if available, is always located below the Overview window on the **ORTI view** menu. Example:



ORTI Tasks

	Name	Events signaled by task	Events task is waiting on	Task that sent message	Priority	Stack save address	Stack starting address	Taskstate	Task starting address	Time rem
	Task1	0x0000	0x0000	NoTask	0x05	0x00021708	0x00021794	IDLE	0x00001648	0x00000
	Task2	0x0000	0x0000	NoTask	0x03	0x00021504	0x00021594	IDLE	0x00001774	0x00000
	Task3	0x0000	0x0000	NoTask	0x06	0x00021304	0x00021394	READY TO RESUME	0x00001828	0x00000
➡	Task4	0x0000	0x0000	NoTask	0x02	0x00000000	0x00021194	RUNNING	0x000018F8	0x00000
	Task5	0x0000	0x0000	NoTask	0x04	0x00020F04	0x00020F94	READY TO RESUME	0x00001918	0x00000
	NO TASK									

The currently active task is indicated by an arrow in the first column, pointing to the task currently running. The last row of the Task window is always NO TASK. When the arrow points to NO TASK execution is taking place outside any task, in the scheduler etc.
The task awareness enables task stepping and task breakpoints, see below.

General Inspection Windows

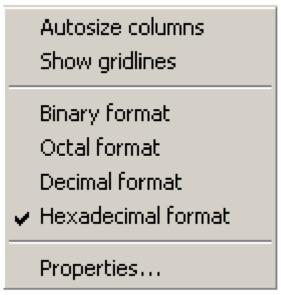
The other windows shown on the **ORTI view** menu are inspection windows of different kind. The number and type of windows depend on the information in the ORTI input file.

Inspection window context menu

The context menu available in an inspection window provides commands to:

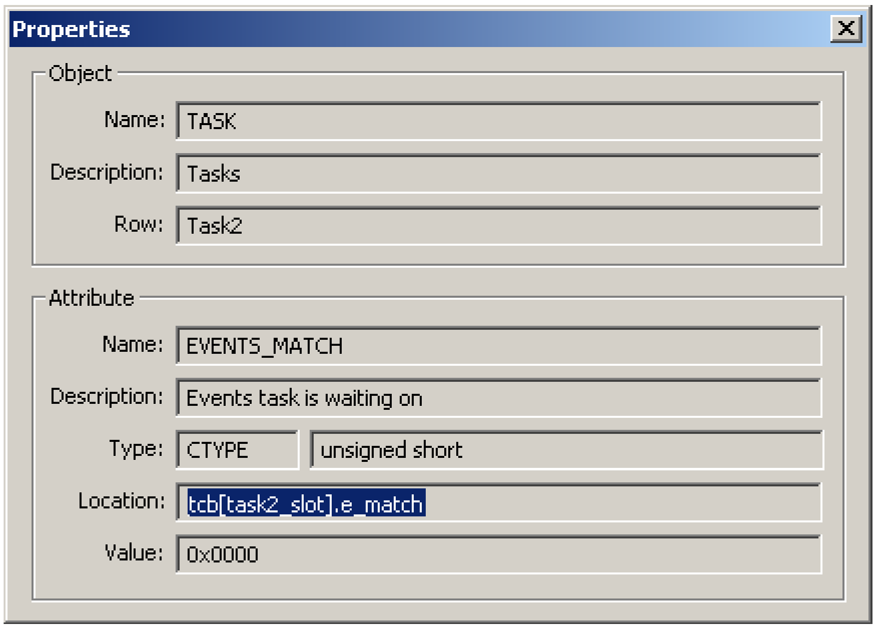
- Automatically resize column widths.
- Toggle gridlines on or off. On is marked with a check mark.
  Note: When gridlines are shown and column widths are resized the window contents may flicker during redraw
- Numeric value display, binary octal, decimal, or hexadecimal. The current display type is marked with a check mark.
- View properties of the item pointed to by the cursor.

.



Properties dialog box

The **Properties** dialog box shows the properties for the selected item. The selected item is framed and marked with red text color in the inspection window.



Properties

Object

Name: TASK

Description: Tasks

Row: Task2

Attribute

Name: EVENTS_MATCH

Description: Events task is waiting on

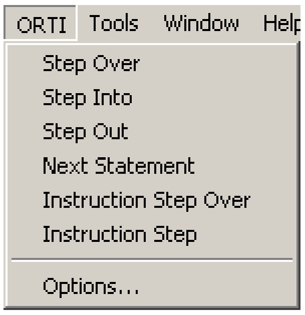
Type: CTYPE unsigned short

Location: tcb[task2_slot].e_match

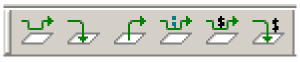
Value: 0x0000

Menu and Toolbar

The **ORTI** menu:



The **ORTI** toolbar:



The menu choices above the separator are the task step items, described in *Task stepping*.

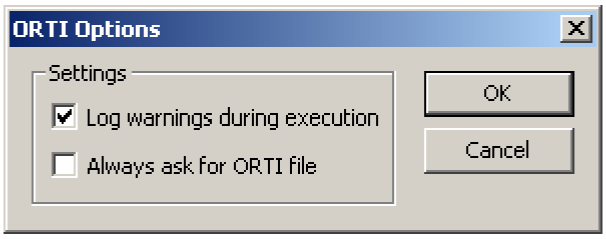
The toolbar buttons correspond to the task step items in the menu.

Note: The menu and toolbar items are enabled when task awareness is available.

Note: In the standard debugger menu, there are no **Instruction Step Over** and **Instruction Step** commands. This is because the standard **Step Over** and **Step Into** commands are context-sensitive, stepping by statement and function call when a source window is active, and stepping by instruction when the disassembly window is active. The RTOS stepping commands are unfortunately *not* context-sensitive; you must choose which kind of step to perform.

Options dialog box

This is the **Options** dialog box:



ORTI Options

Settings

- [✓] Log warnings during execution
- [ ] Always ask for ORTI file

Log warnings during execution

If the driver is unable to read data during application execution at lot of warning messages can be generated in the Debug Log window.
By unchecking the **Log warnings during execution** option, these warning messages are suppressed during execution.

Always ask for ORTI file

By checking the **Always ask for ORTI file** option, the plug-in will always ask for an ORTI file at the session start.

Task stepping

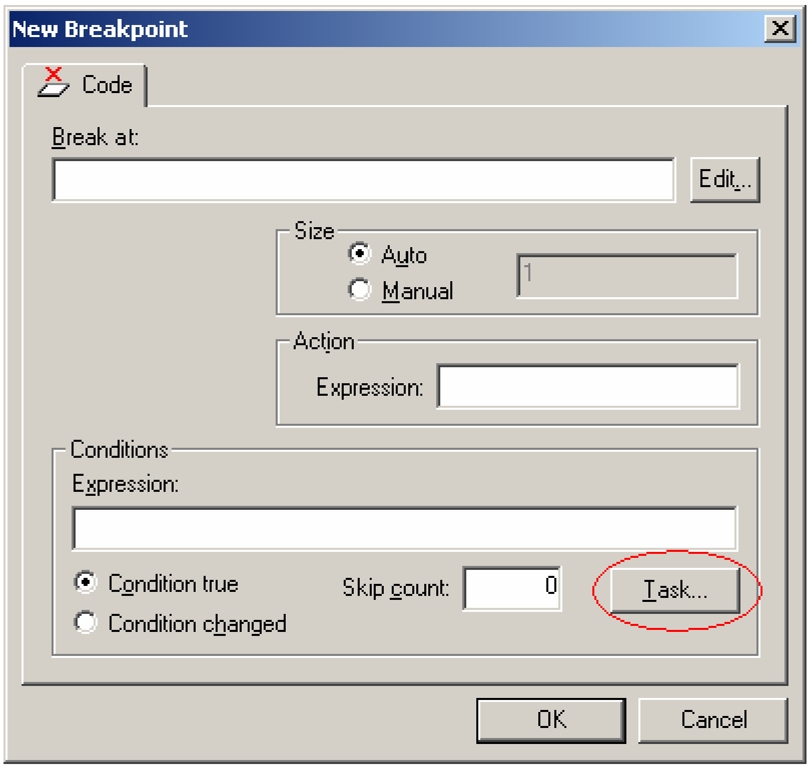
If more than one task can execute the same code, there is a need both for task-specific breakpoints and for task-specific stepping.

For example, consider some utility function, called by several different tasks. Stepping through such a function to verify its correctness can be quite confusing without task-specific stepping. Standard stepping usually works as follows (slightly simplified): When you invoke a step command, the debugger computes one or more locations where that step will end, sets corresponding temporary breakpoints and simply starts execution. When execution hits one of the breakpoints, they are all removed and the step is finished.

Now, during that brief (or not so brief) execution, basically anything can happen in an application with multiple tasks. In particular, a task switch may occur and *another* task may hit one of the breakpoints before the original task does. It may appear that you have performed a normal step, but now you are watching another task. The other task could have called the function with another argument or be in another iteration of a loop, so the values of local variables could be totally different hence, the need for task-specific stepping. The step commands on the **ORTI** menu and in the corresponding toolbar behave just like the normal stepping commands, but they will make sure that the step doesn't finish until the *original* task reaches the step destination.

Breakpoints

The presence of the ORTI plug-in enables a task condition for all standard breakpoints, Code and Data, as shown below:



Break at:

Size: [✓ Auto] Manual 1

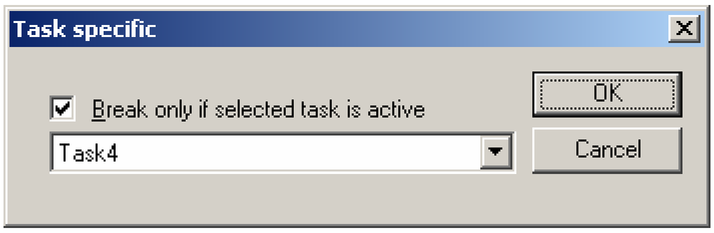
Action Expression:

Conditions Expression:

[✓ Condition true] Condition changed

Skip count: 0

By creating/selecting a Code or Data breakpoint and clicking the **Task...** button the following dialog box will appear:



☑ Break only if selected task is active

Task4

You can make a breakpoint task-specific by clicking the checkbox and selecting a task from the drop-down list.

Note: The drop-down list only shows tasks which have been created at this certain time.

Note: If the code at the breakpoint is only ever executed by one specific task, there is no need to make the breakpoint task-specific.

Note: The **Task…** button is available when task awareness is available.

Trace

SERVICETRACE attribute

If the ORTI description file contains the `OS` object type with the `SERVICETRACE` attribute, the service trace is available. The name of the window and the column name of the service trace attribute is vendor specific.
The service trace display will show the name of the last OS service performed or the name of the service just being executed; in the latter the name will be followed by the string "(executing)", eg: `ActivateTask (executing).`

TOTRACE attribute

Not supported in current version.

Warning and error messages

The ORTI plug-in logs all its warning and error messages in the Debug Log window.
Especially during the startup phase, when the ORTI file is checked, fatal error messages can be displayed.