
TECHNICAL INFORMATION

OSEK operating system

Introduction

This document describes support for OSEK operating system used with winIDEA.

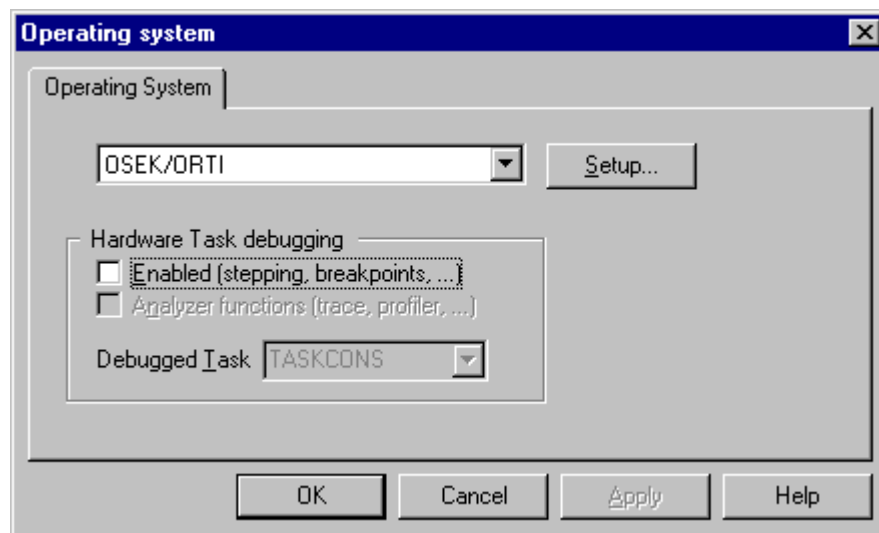
It assumes that:

- Hardware is configured properly,
- An OSEK builder was used to generate the application, with debug support for at least RUNNINGTASK indication. Make sure that a valid ORTI file is generated by entering the appropriate definitions into the OIL file. winIDEA supports ORTI 1.x and 2.x files.
- The OSEK and user sources have been compiled and linked to an executable,
- The executable is listed in the **Debug/Files for download** dialog.

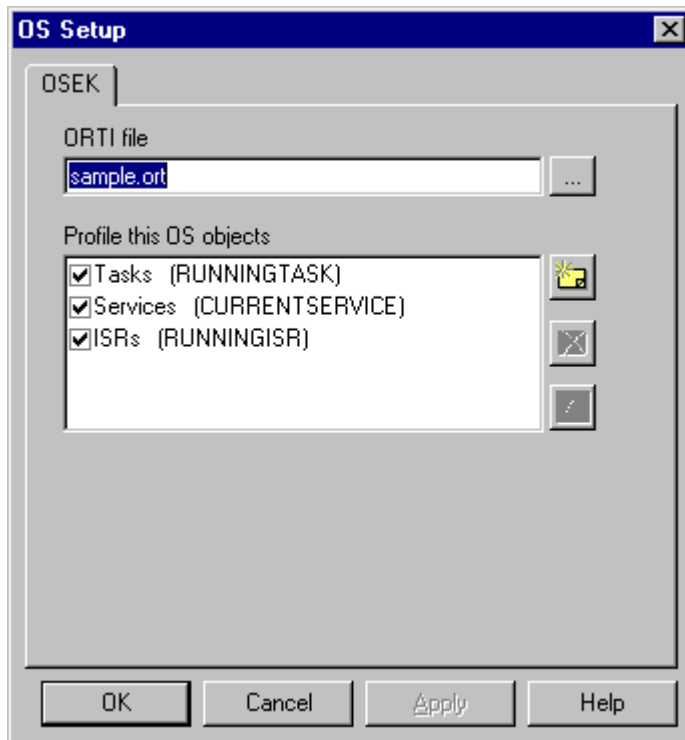
Note that an OSEK Freeze Mode debugging license is required for OSEK debugging.

Configuration

OSEK awareness is configured in the Debug/Operating system dialog:



- Select the **OSEK** option
- Click the **Setup...** button to configure OSEK options

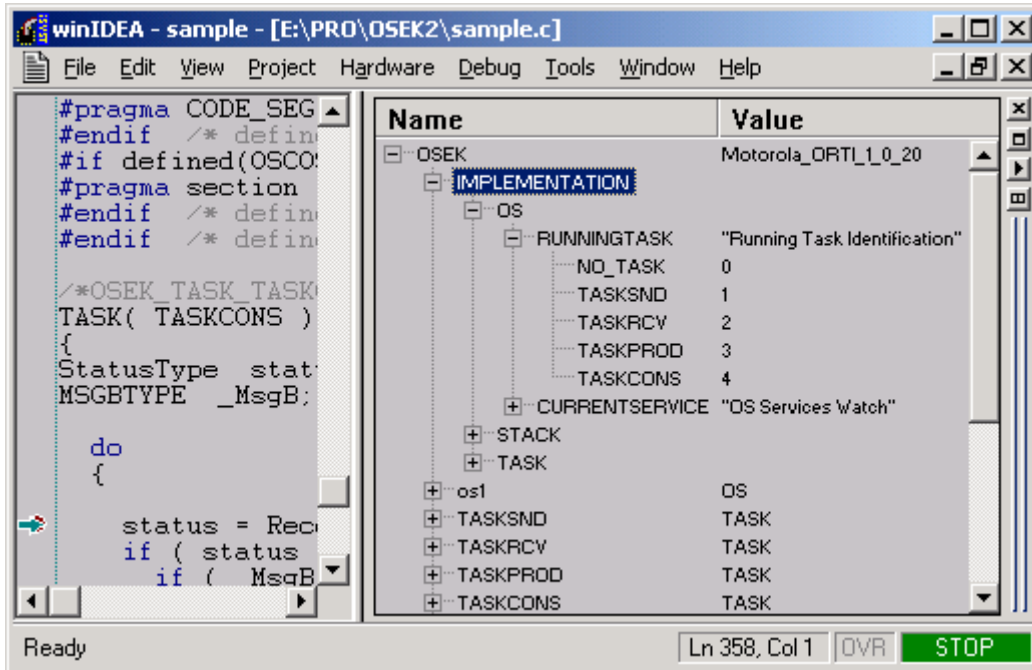


- In the **ORTI file** field specify the path to the ORTI file generated by the OSEK builder.
- Click **OK** in both dialogs to confirm configuration
- Perform **Debug/Download**

Viewing OSEK objects

OSEK objects are displayed in the OS window

- Select the **View/Operating System** menu item
- The top item in the OS window displays the operating system (OSEK), value displays the version as reported in the ORTI file.



IMPLEMENTATION

The item **IMPLEMENTATION** is special as it defines the layout of other display OSEK objects. In the figure above, it defines layout of **OS**, **STACK** and **TASK** objects.

The **OS** object layout (expanded) defines two members: **RUNNINGTASK** and **CURRENTSERVICE**. Every member has a description identifying its meaning, displayed in the **Value** column.

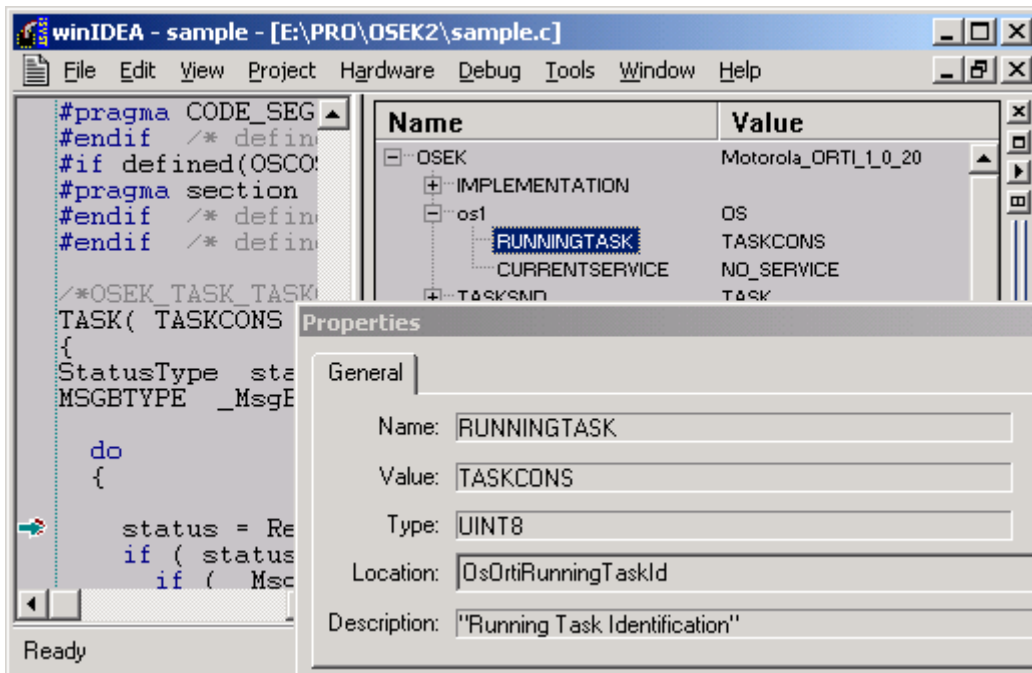
When an OS object member is expanded, it shows valid values (if applicable) for this member to assume. In the above figure, the **RUNNINGTASK** can assume values 0 through 4. When value 0 is contained, **NO_TASK** task is executing, value 1 refers to **TASKSND** etc.

Note: the IMPLEMENTATION

Objects

There can be one or more objects of a type defined in the **IMPLEMENTATION** part. In this example, one OS type object, 5 TASK and 9 STACK objects exist.

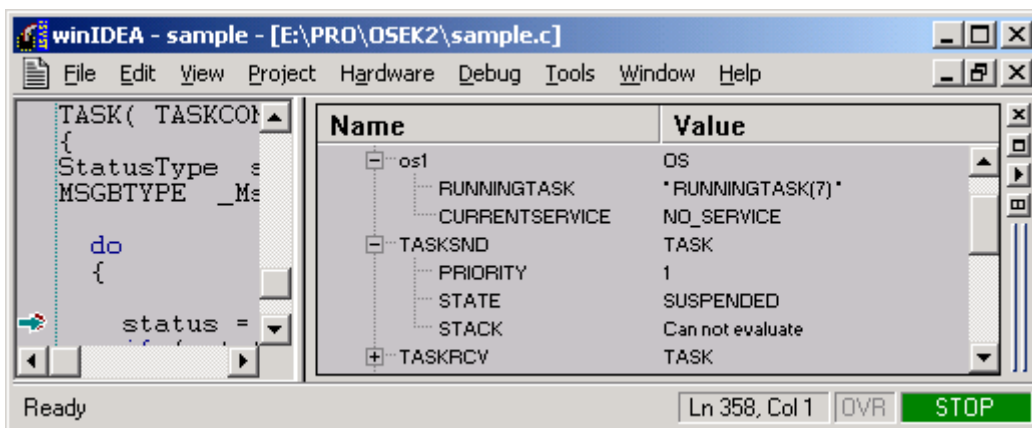
When an OS object is expanded it will shown current values for its members.



In the above figure, the **os1** object (of type **OS**), shows current values for **RUNNINGTASK** and **CURRENTSERVICE**. winIDEA reads out its values and maps them to the definition table (**IMPLEMENTATION/OS/RUNNINGTASK**, **IMPLEMENTATION/OS/CURRENTSERVICE**).

There are two situations where a value of an OS object member can not be displayed.

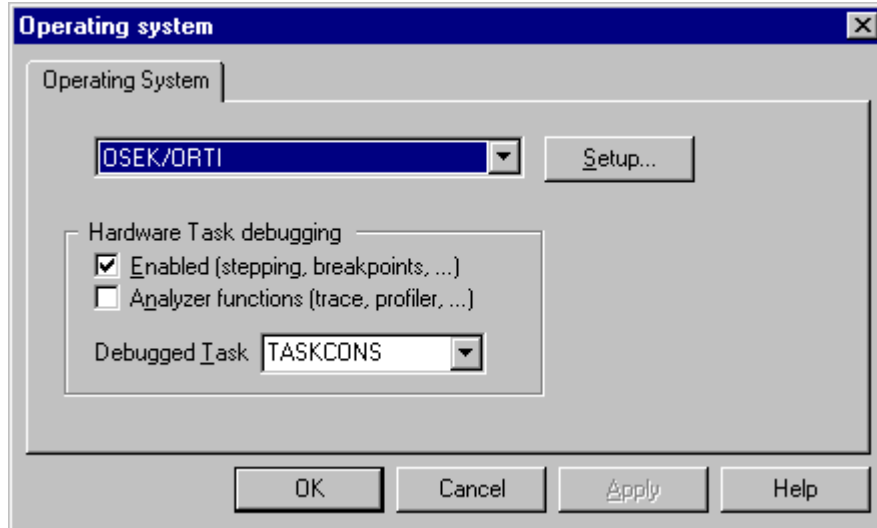
- If the member contains value that can not be found in the definition table (which should only occur before OS initialization starts). See **RUNNINGTASK** in below figure.
- If the expression, by which the value is calculated can not be evaluated. In the bellow figure the **STACK** of **TASKSND** task is not allocated as the task is suspended.



Task debugging

To enable task debugging:

- Check the **Enabled** option in the **Debug/Operating System** dialog



- Select the task to be debugged.
- If you wish to perform task specific analyzer operations, check the **Analyzer functions** option. In this case the trace and the profiler will record only objects in the selected task. In the above example, if the *Analyzer functions* options would be checked, only functions within the **TASKCONS** function will be traced or profiled. When all tasks should be either traced or profiled, the option must be unchecked.

In the above configuration, breakpoints that occur in the context of task **TASKCONS** stop the execution only.

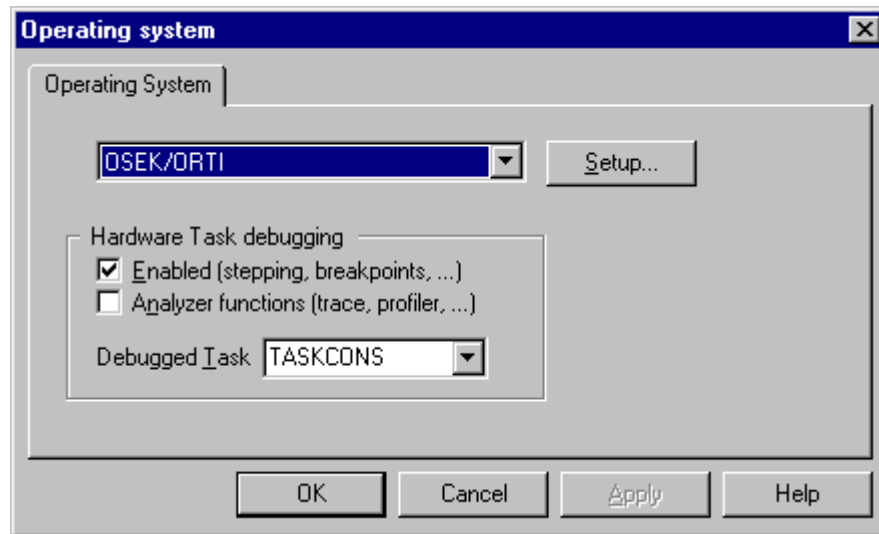
Note that task debugging must be supported by the hardware.

Application Notes

When using OSEK operating system, it must be selected in the *Operating System* dialog first. Note that associated software license must be obtained, when the debugger's OS awareness is required.

Task Debugging

When using operating system, the debugging can be limited to one task only and the application can be debugged task-by-task. Tasks, other than the selected one, don't interfere with the selected task. To enable task debugging, the *Enabled* checkbox must be checked and the particular task (e.g. TASKCONS) being debugged selected in the *Debug/Operating System* dialog.



Operating System dialog

Typically, the program contains many functions, which are called from different tasks. With no task debugging, if a breakpoint is set in the function, every time a call is made to the function and the breakpoint is reached, the program execution stops. When the task debugging is enabled, the program execution stops only when the function is called from the task (e.g. TASKCONS) specified in the *Debug/Operating System* dialog. When any other task calls the function with the breakpoint set, the program execution does not stop.

When task debugging is enabled and particular task selected, breakpoints and debug commands like *step*, *step over*, don't function in tasks, other than the selected one. Have in mind that *step* and *step over* debug commands are implemented using *run* debug command and breakpoints

Analyzer functions

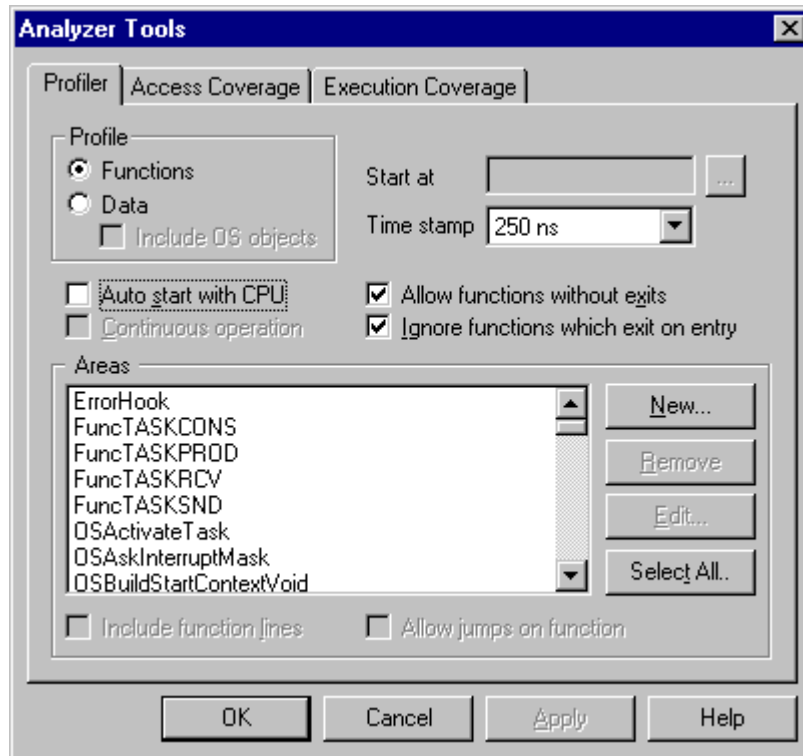
Some analyzer operating modes like profiler and trace can be limited to the specific task. Additionally, the profiler can profile either functions or the operating system objects.

Trace

When the *Analyzer functions* checkbox is not checked in the *Operating system* dialog, the trace doesn't feature any task awareness. When the *Analyzer functions* checkbox is checked and a particular task selected, the trace records only the CPU cycles while the selected task is being active. Additionally, when any trigger is set, it is evaluated only when the selected task is active.

Profiling functions

When profiling functions, called by the particular task only, the *Analyzer functions* option must be checked and a task selected from the combo box. When the option is unchecked, all functions can be profiled regardless to the active task. Refer to the profiler user's guide for more details on profiling functions. Make sure that the function profiler is selected in the Debug/Analyzer Tools/Profiler dialog is selected.



Profiler Configuration Dialog

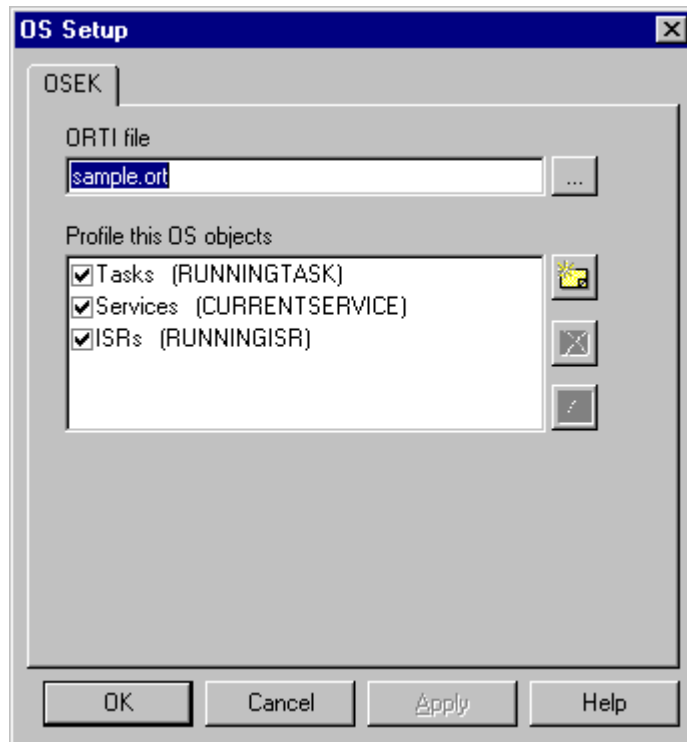
Profiling OS objects

The OS profiler can be used to record task and service activity.

Note: To get correct results Task Debugging should be disabled when profiling OS objects.

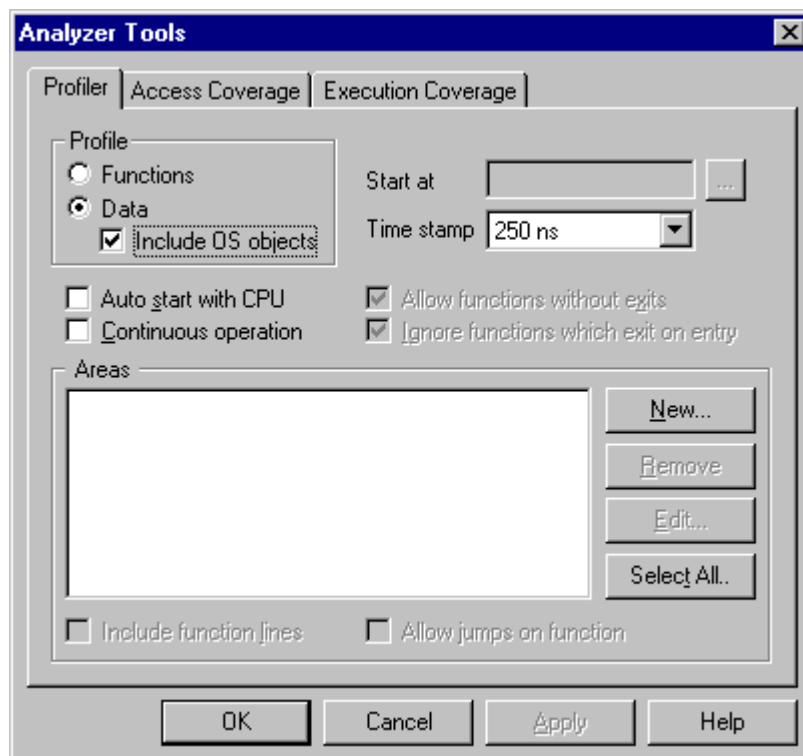
To enable OS profiler:

- Configure the analyzer to operate in profiler mode (**Hardware/Analyzer Setup/Operation**)
- Invoke the *OS Setup* dialog from the **Debug/Operating System** dialog to profile the OS objects. Profiled OS objects must be selected. Note that this example is OSEK based.



OSEK Setup dialog

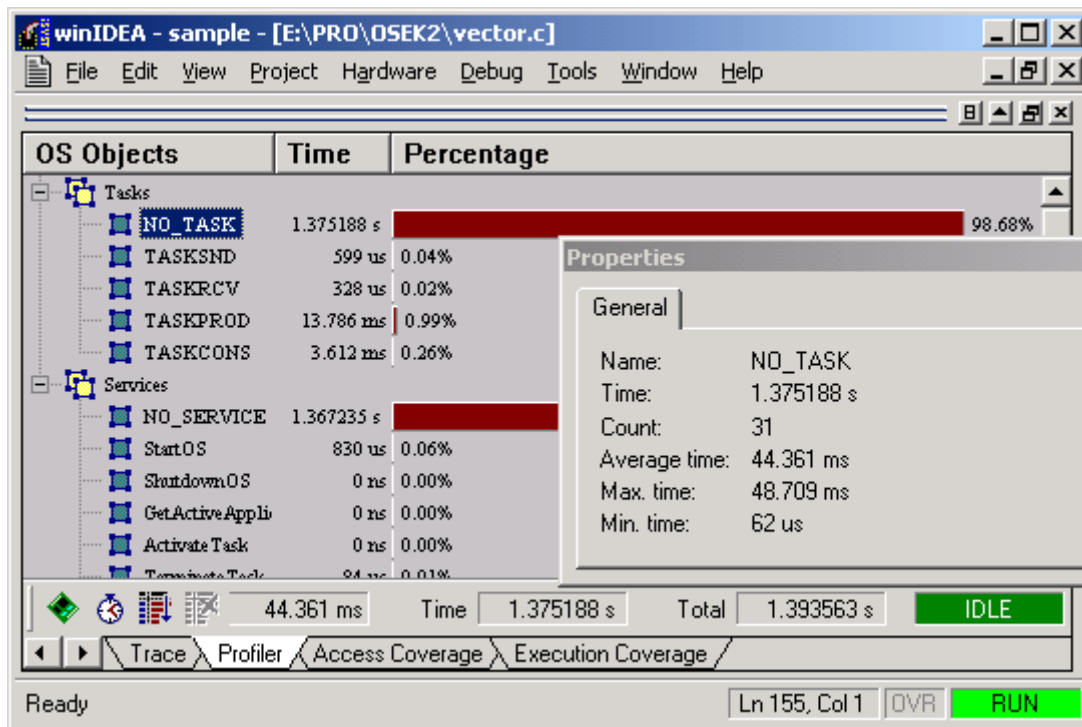
Next, make sure the profiler is set to profile Data and that the *Include OS objects* option is selected in the profiler configuration dialog.



Profiler configuration dialog

- Open the Profiler/Coverage window (**View/Profiler/Coverage**)

- Start the profiler
- Select **OS Statistics** from the context menu to view time and activation statistics. Individual object's statistics can be inspected by selecting Properties command from the context menu.



- Select **OS Execution** from the context menu to view execution history.

