

Using trace

This document replaces the corresponding information in the IAR Embedded Workbench® IDE User Guide for ARM®.

This chapter gives you information about collecting and using trace data, reference information for trace-related windows and dialog boxes.

Collecting and using trace data

Before you start using the trace-related functionality in C-SPY, read this section to get a good understanding of how to take advantage of the trace data for debugging.

More specifically, these topics are covered:

- *Reasons for using trace*, page 2
- *Briefly about trace*, page 2
- *Requirements for using trace*, page 3
- *Getting started with trace in the C-SPY simulator*, page 4
- *Getting started with ETM trace*, page 4
- *Getting started with SWO trace*, page 5
- *Setting up concurrent use of ETM and SWO*, page 5
- *Trace data collection using breakpoints*, page 6
- *Searching in trace data*, page 6
- *Browsing through trace data*, page 7.

When you need detailed information about a specific dialog box or window, see *Trace-related reference information*, page 7.

For related information, see also:

- *Using J-Link trace triggers and trace filters* (in *IAR Embedded Workbench® IDE User Guide for ARM®*)
- *Using the data and interrupt logging system* (in *IAR Embedded Workbench® IDE User Guide for ARM®*)
- *Using the profiler*, page 35.

REASONS FOR USING TRACE

By using trace, you can inspect the program flow up to a specific state, for instance an application crash, and use the trace data to locate the origin of the problem. Trace data can be useful for locating programming errors that have irregular symptoms and occur sporadically.

BRIEFLY ABOUT TRACE

Your target system must be able to generate trace data. Once generated, C-SPY can collect it and you can visualize and analyze the data in various windows and dialog boxes.

C-SPY supports collecting trace data from these target systems:

- Devices with support for ETM (Embedded Trace Macrocell)—ETM trace
- Devices with support for the SWD (Serial Wire Debug) interface using the SWO (Serial Wire Output) communication channel—SWO trace
- The C-SPY simulator.

Depending on your target system, different types of trace data can be generated.

ETM trace

ETM trace (also known as full trace) is a continuously collected sequence of every executed instruction for a selected portion of the execution. Usually, it is not possible to collect very long sequences of data in real time without saturating the communication channel that transmits the data to C-SPY.

The debug probe contains a trace buffer that collects trace data in real time, but the data is not displayed in the C-SPY windows until after the execution has stopped.

SWO trace

SWO trace is a sequence of events of various kinds, generated by the on-chip debug hardware. The events are transmitted in real time from the target system over the SWO communication channel. This means that the C-SPY windows are continuously updated while the target system is executing. The most important events are:

- PC sampling

The hardware can sample and transmit the value of the program counter at regular intervals. This is not a continuous sequence of executed instructions (like ETM trace), but a sparse regular sampling of the PC. A modern ARM CPU typically executes millions of instructions per second, while the PC sampling rate is usually counted in thousands per second.

- Interrupt logs

The hardware can generate and transmit data related to the execution of interrupts, generating events when entering and leaving an interrupt handler routine.

- Data logs

Using Data Log breakpoints, the hardware can be configured to generate and transmit events whenever a certain variable, or simply an address range, is accessed by the CPU.

The SWO channel does not have unlimited throughput, so it is usually not possible to use all the above features at the same time, at least not if either the frequency of PC sampling, of interrupts, or of accesses to the designated variables is high.

Trace-related features in C-SPY

In C-SPY, you can use the trace-related windows Trace, Function Trace, Timeline, and Find in Trace. In the C-SPY simulator, you can also use the Trace Expressions window. Depending on your C-SPY driver, you can set various types of trace breakpoints and triggers to control the collection of trace data.

In addition, several other features in C-SPY also use trace data, features such as the Profiler, Code coverage, and Instruction profiling.

If you use the C-SPY J-Link/J-Trace driver, you have access to windows such as the Interrupt Log, Interrupt Log Summary, Data Log, and Data Log Summary windows.



When you are debugging, two buttons labeled **ETM** and **SWO**, respectively, are visible on the IDE main window toolbar. If any of these buttons is green, it means that the corresponding trace hardware is generating trace data. Just point at the button with the mouse pointer to get detailed tooltip information about which C-SPY features that have requested trace data generation. This is useful, for example, if your SWO communication channel often overflows because too many of the C-SPY features are currently using trace data. Clicking on the buttons opens the corresponding setup dialog boxes.

REQUIREMENTS FOR USING TRACE

To use trace-related functionality in C-SPY, you need debug components (hardware, a debug probe, and a C-SPY driver) that all support trace. Alternatively, you can use the trace features provided by the C-SPY simulator.

Note: The specific set of debug components you are using determine which of the trace features in C-SPY that are supported.

Requirements for using ETM trace

ETM trace is available for some ARM devices.

To use ETM trace you need one of these combinations:

- A J-Trace debug probe and a device that supports ETM. Make sure to use the C-SPY J-Link/J-Trace driver.
- A J-Link debug probe and a device that supports ETM with ETB (Embedded Trace Buffer). Make sure to use the C-SPY J-Link/J-Trace driver.
- The C-SPY RDI driver, and a debug probe and a device that both support ETM.

Requirements for using SWO trace

To use SWO trace you need a J-Link or J-Trace debug probe that supports the SWO communication channel and a device that supports the SWD/SWO interface.

GETTING STARTED WITH TRACE IN THE C-SPY SIMULATOR

To collect trace data using the C-SPY simulator, no specific build settings are required.



- 1 After you have built your application and started C-SPY, choose **Simulator>Trace** to open the Trace window, and click the **Activate** button to enable collecting trace data.
- 2 Start the execution. When the execution stops, for instance because a breakpoint is triggered, trace data is displayed in the Trace window. For more information about the window, see *Trace window*, page 15.

GETTING STARTED WITH ETM TRACE

To set up ETM trace, follow these steps:

- 1 Before you start C-SPY:
 - For the C-SPY RDI driver, choose **Project>Options>RDI>ETM trace**
 - For J-Trace no specific settings are required before starting C-SPY
 - The trace port must be set up. For some devices this is done automatically when the trace logic is enabled. However, for some devices, typically Atmel and ST devices based on ARM 7 or ARM 9, you need to set up the trace port explicitly. You do this by means of a C-SPY macro file. You can find examples of such files (`ETM_init*.mac`) in the example projects. To use a macro file, choose **Project>Options>Debugger>Setup>Use macro files(s)**. Specify your macro file; a browse button is available for your convenience.

Note that the pins used on the hardware for the trace signals cannot be used by your application.

- 2 After you have started C-SPY, choose **ETM Trace Settings** from the driver-specific menu on the menu bar. In the **ETM Trace Settings** dialog box that appears, check if you need to change any of the default settings. For details, see *ETM Trace Settings dialog box*, page 8.
-  3 Open the Trace window—available from the driver-specific menu—and click the **Activate** button to enable trace data collection.
- 4 Start the execution. When the execution stops, for instance because a breakpoint is triggered, trace data is displayed in the Trace window. For more information about the window, see *Trace window*, page 15.

GETTING STARTED WITH SWO TRACE

To set up SWO trace, follow these steps:

- 1 Before you start C-SPY, choose **Project>Options>J-Link/J-Trace** and click the **Connection** tab. Choose **Interface>SWD**.
- 2 After you have started C-SPY, choose **SWO Trace Windows Settings** from the **J-Link** menu. In the **SWO Trace Windows Settings** dialog box that appears, make your settings for controlling the output in the Trace window. For details, see *SWO Trace Window Settings dialog box*, page 10.
- 3 To configure the hardware's generation of trace data, click the **SWO Configuration** button available in the **SWO Configuration** dialog box. For details, see *SWO Configuration dialog box*, page 12.



Note specifically these settings:

- The value of the **CPU clock** option must reflect the frequency of the CPU clock speed at which the application executes. Note also that the settings you make are preserved between debug sessions.
- To decrease the amount of transmissions on the communication channel, you can disable the **Timestamp** option. Alternatively, set a lower rate for PC Sampling or use a higher SWO clock frequency.

-  4 Open the Trace window—available from the **J-Link/J-Trace** menu—and click the **Activate** button to enable trace data collection.
- 5 Start the execution. The Trace window is continuously updated with trace data. For more information about this window, see *Trace window*, page 15.

SETTING UP CONCURRENT USE OF ETM AND SWO

If you have a J-Trace debug probe for Cortex-M3, you can use ETM trace and SWO trace concurrently.

In this case, if you activate the ETM trace and the SWO trace, SWO trace data will also be collected in the ETM trace buffer, instead of being streamed via the SWO channel. This means that the SWO trace data will not be displayed until the execution has stopped, instead of being continuously updated live in the Trace window.

TRACE DATA COLLECTION USING BREAKPOINTS

A convenient way to collect trace data between two execution points is to start and stop the data collection using dedicated breakpoints. In the editor or Disassembly window, position your insertion point, right-click, and toggle a **Trace Start** or **Trace Stop** breakpoint from the context menu. Alternatively, in the Breakpoints window, choose **Trace Start**, **Trace Stop**, or **Trace Filter**. In the C-SPY simulator, the C-SPY system macros `__setTraceStartBreak` and `__setTraceStopBreak` can also be used.

For details about these breakpoints, see *Trace Start breakpoints dialog box*, page 29 and *Trace Stop breakpoints dialog box*, page 30, respectively.

SEARCHING IN TRACE DATA

When you have collected trace data, you can perform searches in the collected data to locate the parts of your code or data that you are interested in, for example, a specific interrupt or accesses of a specific variable.

You specify the search criteria in the **Find in Trace** dialog box and view the result in the Find in Trace window.

The Find in Trace window is very similar to the Trace window, showing the same columns and data, but *only* those rows that match the specified search criteria.

Double-clicking an item in the Find in Trace window brings up the same item in the Trace window.

To search in your trace data, follow these steps:



1 In the Trace window toolbar, click the **Find** button.

2 In the **Find in Trace** dialog box, specify your search criteria.

Typically, you can select to search for:

- A specific piece of text, for which you can apply further search criteria
- An address range
- A combination of these, like a specific piece of text within a specific address range.

For detailed information about the different options, see *Find in Trace dialog box*, page 32.

- 3 When you have specified your search criteria, click **Find**. The Find in Trace window is displayed, which means you can start analyzing the trace data. For detailed reference information, see *Find in Trace window*, page 33.

BROWSING THROUGH TRACE DATA

To follow the execution history, simply look and scroll in the Trace window. Alternatively, you can enter *browse mode*.



To enter browse mode, double-click an item in the Trace window, or click the **Browse** toolbar button.

The selected item turns yellow and the source and disassembly windows will highlight the corresponding location. You can now move around in the trace data using the up and down arrow keys, or by scrolling and clicking; the source and Disassembly windows will be updated to show the corresponding location. This is like stepping backward and forward through the execution history.

Double-click again to leave browse mode.

Trace-related reference information

To use the trace system, you might need reference information about these windows and dialog boxes:

- *ETM Trace Settings dialog box*, page 8
- *SWO Trace Window Settings dialog box*, page 10
- *SWO Configuration dialog box*, page 12
- *Trace window*, page 15 (ETM Trace, SWO Trace, and Trace for the C-SPY simulator)
- *Function Trace window*, page 20
- *Timeline window*, page 21
- *Timeline window*, page 21
- *Viewing Range dialog box*, page 28
- *Trace Start breakpoints dialog box*, page 29 (for the C-SPY simulator only)
- *Trace Start breakpoints dialog box* for the C-SPY J-Link/J-Trace driver only (in *IAR Embedded Workbench® IDE User Guide for ARM®*)
- *Trace Stop breakpoints dialog box*, page 30 (for the C-SPY simulator only)
- *Trace Stop breakpoints dialog box* for the C-SPY J-Link/J-Trace driver only (in *IAR Embedded Workbench® IDE User Guide for ARM®*)

- Trace Filter breakpoints dialog box for the C-SPY J-Link/J-Trace driver only (in *IAR Embedded Workbench® IDE User Guide for ARM®*)
- *Trace Expressions window*, page 31 (for the C-SPY simulator only)
- *Find in Trace window*, page 33
- *Find in Trace dialog box*, page 32.

ETM TRACE SETTINGS DIALOG BOX

The **ETM Trace Settings** dialog box is available from the driver-specific menu.

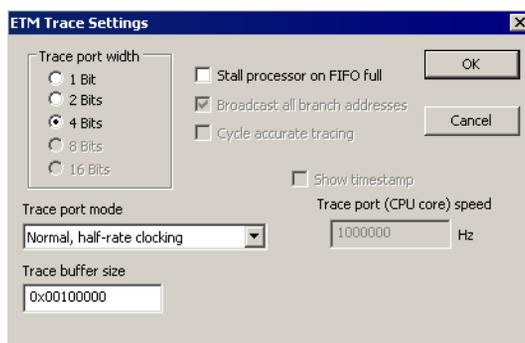


Figure 1: ETM Trace Settings dialog box

Note: This dialog box looks slightly different for the RDI drivers.

Using this dialog box

Use the **ETM Trace Settings** dialog box to configure ETM trace generation and collection.

See also *Getting started with ETM trace*, page 4.

Trace port width

The trace bus width can be set to 1, 2, 4, 8, or 16 bits. The value must correspond with what is supported by the hardware and the debug probe. For Cortex-M3, 1, 2, and 4 bits are supported by the J-Trace debug probe. For ARM7/9, only 4 bits are supported by the J-Trace debug probe.

Trace port mode

Use these options to set the trace clock rate; choose between:

- Normal, full-rate clocking

- Normal, half-rate clocking
- Multiplexed
- Demultiplexed
- Demultiplexed, half-rate clocking.

Note: For RDI drivers, only the two first alternatives are available. For the J-Trace driver, the available alternatives depend on the device you are using.

Trace buffer size

Use the text box to specify the size of the trace buffer. By default, the number of trace frames is 0x10000. For ARM7/9 the maximum number is 0x100000, and for Cortex-M3 the maximum number is 0x400000.

For ARM7/9, one trace frame corresponds to 2 bytes of the physical J-Trace buffer size. For Cortex-M3, one trace frame corresponds to approximately 1 byte of the buffer size.

Note: The **Trace buffer size** option is only available for the J-Trace driver.

Cycle accurate tracing

Select this option to emit trace frames synchronous to the processor clock even when no trace data is available. This makes it possible to use the trace data for real-time timing calculations. However, if you select this option, the risk for FIFO buffer overflow increases.

Note: The **Cycle accurate tracing** option is only available for ARM7/9 devices.

Broadcast all branch addresses

Use this option to make the processor send more detailed address trace information. However, if you select this option, the risk for FIFO buffer overflow increases.

Note: The **Broadcast all branch addresses** option is only available for ARM7/9 devices. For Cortex, this option is always enabled.

Stall processor on FIFO full

The trace FIFO buffer might in some situations become full—FIFO buffer overflow—which means trace data will be lost. If you use this option, the processor will be stalled in case the FIFO buffer fills up.

Show timestamp

Use this option to make the Trace window display seconds instead of cycles in the **Index** column. To make this possible you must also specify the appropriate speed for your CPU in the **Trace port (CPU core) speed** text box.

Note: The **Show timestamp** option is only available when you use the J-Trace driver with ARM7/9 devices.

SWO TRACE WINDOW SETTINGS DIALOG BOX

The **SWO Trace Window Settings** dialog box is available from the **J-Link** menu, alternatively from the SWO Trace window toolbar.

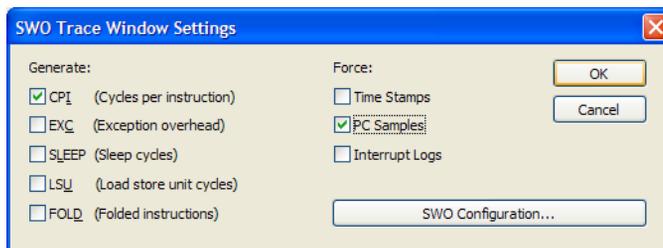


Figure 2: SWO Trace Window Settings dialog box

Using this dialog box

Use the **SWO Trace Window Settings** dialog box to specify what you want to display in the Trace window.

Note that you also need to configure the generation of trace data, click **SWO Configuration**. For more information, see *SWO Configuration dialog box*, page 12.

Force

Use this option to enable trace data generation, if it is not already enabled by other features using SWO trace data. The Trace window displays all generated SWO data. Other features in C-SPY, for example Profiling, can also enable SWO trace data generation. If no other feature has enabled the generation, use the **Force** options to generate SWO trace data.

The generated data will be displayed in the Trace window. Choose between:

Time Stamps

Enables timestamps for various SWO trace packets, that is sent over the SWO communication channel. Use the resolution drop-down list to choose the resolution of the timestamp value. For example, 1 to count every cycle, or 16 to count every 16th cycle. Note that the lowest resolution is only useful if the time between each event packet is long enough. 16 is useful if using a low SWO clock frequency.

PC samples	Enables sampling the program counter register, PC, at regular intervals. To choose the sampling rate, see <i>PC Sampling</i> , page 13.
Interrupt Logs	Enables generation of interrupt logs. For information about other C-SPY features that also use trace data for interrupts, see <i>Using the data and interrupt logging system (in IAR Embedded Workbench® IDE User Guide for ARM®)</i> .

Generate

Use this option to enable trace data generation for these events. The generated data will be displayed in the Trace window. The value of the counters are displayed in the **Comment** column in the SWO Trace window. Choose between:

CPI	Enables generation of trace data for the CPI counter.
EXC	Enables generation of trace data for the EXC counter.
SLEEP	Enables generation of trace data for the SLEEP counter.
LSU	Enables generation of trace data for the LSU counter.
FOLD	Enables generation of trace data for the FOLD counter.

SWO Configuration button

Displays the **SWO Configuration** dialog box where you can configure the hardware's generation of trace data. See *SWO Configuration dialog box*, page 12.

SWO CONFIGURATION DIALOG BOX

The **SWO Configuration** dialog box is available from the **J-Link** menu, alternatively from the **SWO Trace Window Settings** dialog box.

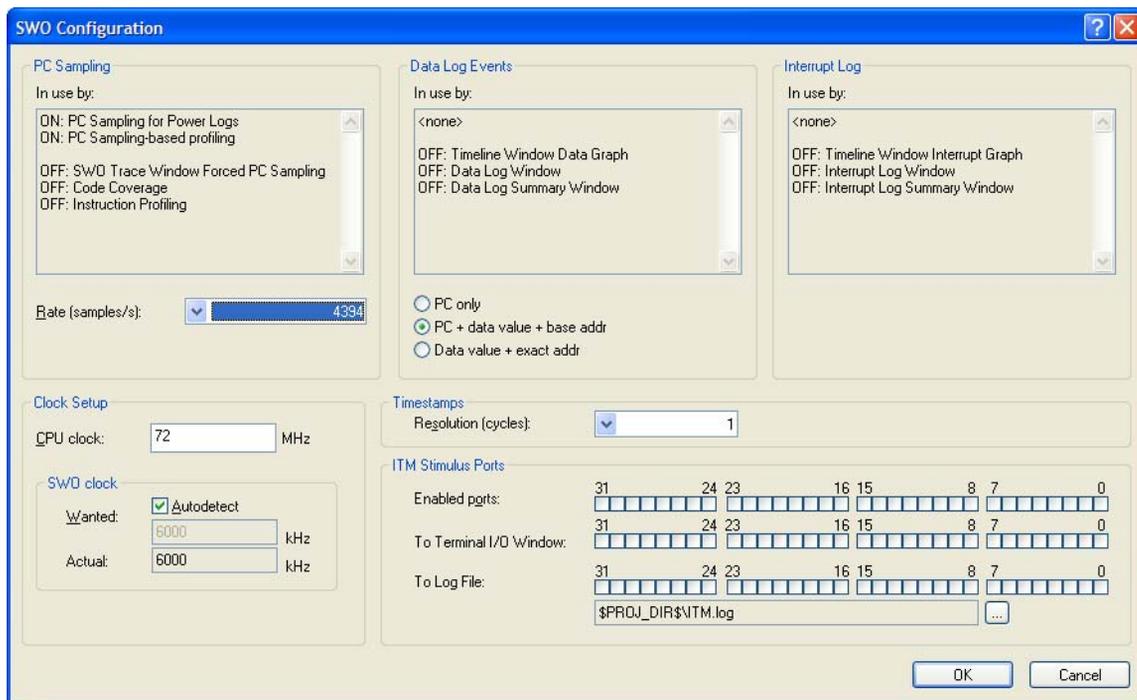


Figure 3: SWO Configuration dialog box

Using this dialog box

Use the **SWO Configuration** dialog box to configure the serial-wire output communication channel and the hardware's generation of trace data.

See also *Getting started with SWO trace*, page 5.

PC Sampling

These items are available:

In use by	Lists the features in C-SPY that can use trace data for PC Sampling. ON indicates features currently using trace data. OFF indicates features currently not using trace data.
Rate	Use the drop-down list to choose the sampling rate, that is, the number of samples per second. The highest possible sampling rate depends on the SWO clock value and on how much other data that is sent over the SWO communication channel. The higher values in the list will not work if the SWO communication channel is not fast enough to handle that much data.

Data Log Events

Use this option to specify what to log when a Data Log breakpoint is triggered. These items are available:

In use by	Lists the features in C-SPY that can use trace data for Data Log Events. ON indicates features currently using trace data. OFF indicates features currently not using trace data.
PC only	Logs the value of the program counter.
PC + data value + base addr	Logs the value of the program counter, the value of the data object, and its base address.
Data value + exact addr	Logs the value of the data object and the exact address of the data object that was accessed.

Interrupt Log

Interrupt Log lists the features in C-SPY that can use trace data for Interrupt Logs. ON indicates features currently using trace data. OFF indicates features currently not using trace data.

For more information about interrupt logging, see *Using the data and interrupt logging system* (in *IAR Embedded Workbench® IDE User Guide for ARM®*).

CPU clock

Use this option to specify the exact clock frequency used by the internal processor clock, `HCLK`, in MHz. The value can have decimals.

This value is used for configuring the SWO communication speed and for calculating time stamps.

SWO clock

Use this option to specify the clock frequency of the SWO communication channel in kHz. Choose between these options:

Autodetect	Automatically uses the highest possible frequency that the J-Link debug probe can handle.
Wanted	Manually selects the frequency to be used. The value can have decimals. Use this option if data packets are lost during transmission.

The clock frequency that is actually used is displayed in the **Actual** text box.

Timestamps

Use this drop-down list to choose the resolution of the timestamp value. For example, 1 to count every cycle, or 16 to count every 16th cycle. Note that the lowest resolution is only useful if the time between each event packet is long enough.

ITM Stimulus Ports

The ITM Stimulus Ports are used for sending data from your application to the debugger host without stopping the program execution. There are 32 such ports.

Use the check boxes to select which ports you want to redirect and to where:

- Enabled ports** Enables the ports to be used. Only enabled ports will actually send any data over the SWO communication channel to the debugger.
- To Terminal I/O window** Specifies the ports to use for routing data to the Terminal I/O window.
- To Log File** Specifies the ports to use for routing data to a log file. To use a different log file than the default one, use the browse button.



The `stdout` and `stderr` of your application can be rerouted via SWO and this means that `stdout/stderr` will appear in the C-SPY Terminal I/O window. To achieve this, choose **Project>Options>General Options>Library Configuration>Library low-level interface implementation>stdout/stderr>Via SWO**.

This can be disabled if you deselect the port settings in the **Enabled ports** and **To Terminal I/O** options.

TRACE WINDOW

The Trace window is available from the driver-specific menu.

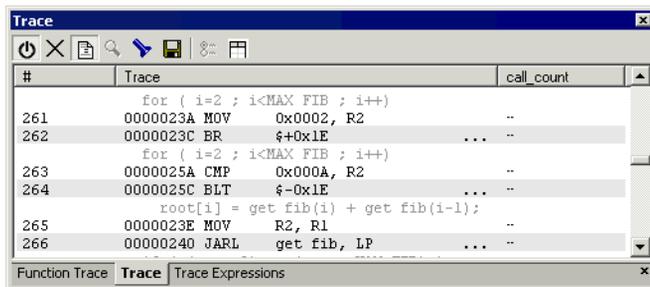


Figure 4: The Trace window in the simulator

Note: There are three different Trace windows—ETM Trace, SWO Trace, and just Trace for the C-SPY simulator. The windows look slightly different.

Using this window

The Trace window displays the collected trace data, where the content differs depending on the C-SPY driver you are using and the trace support of your debug probe:

C-SPY simulator	The window displays a collected sequence of executed machine instructions. In addition, the window can display trace data for expressions.
ETM trace	The window displays the sequence of executed instructions—optionally with embedded source—which has been continuously collected during application execution, that is <i>full trace</i> . The data has been collected in the ETM trace buffer. The collected data is displayed after the execution has stopped. For information about the requirements for using ETM trace, see <i>Requirements for using ETM trace</i> , page 4.
SWO trace	The window displays all events transmitted on the SWO channel. The data is streamed from the target system, via the SWO communication channel, and continuously updated live in the Trace window. Note that if you use the SWO communication channel on a trace probe, the data will be collected in the trace buffer and displayed after the execution has stopped. For information about the requirements for using SWO trace, see <i>Requirements for using SWO trace</i> , page 4.

Trace toolbar

The Trace toolbar at the top of the Trace window and in the Function trace window provides these toolbar buttons:

Toolbar button	Description
	Enable/Disable
	Clear trace data
	Toggle Source
	Browse

Table 1: Trace toolbar buttons

Toolbar button	Description
 Find	Displays the Find in Trace dialog box where you can perform a search; see <i>Find in Trace dialog box</i> , page 32.
 Save	In the ETM Trace and SWO Trace windows this button displays the Trace Save dialog box, see <i>Trace Save dialog box</i> , page 19. In the C-SPY simulator this button displays a standard Save As dialog box where you can save the collected trace data to a text file, with tab-separated columns.
 Edit Settings	In the ETM Trace window this button displays the Trace Settings dialog box, see <i>ETM Trace Settings dialog box</i> , page 8. In the SWO Trace window this buttons displays the SWO Trace Window Settings dialog box, see <i>SWO Trace Window Settings dialog box</i> , page 10. In the C-SPY RDI driver this button is not available. In the C-SPY simulator this button is not enabled.
 Edit Expressions (C-SPY simulator only)	Displays the Trace Expressions window; see <i>Trace Expressions window</i> , page 31.

Table 1: Trace toolbar buttons (Continued)

Trace display area in the C-SPY simulator

The Trace window contains the following columns for the C-SPY simulator:

Trace window column	Description
#	A serial number for each row in the trace buffer. Simplifies the navigation within the buffer.
Cycles	The number of cycles elapsed to this point.
Trace	The collected sequence of executed machine instructions. Optionally, the corresponding source code can also be displayed.
<i>Expression</i>	Each expression you have defined to be displayed appears in a separate column. Each entry in the expression column displays the value <i>after</i> executing the instruction on the same row. You specify the expressions for which you want to collect trace data in the Trace Expressions window; see <i>Trace Expressions window</i> , page 31.

Table 2: Trace window columns for the C-SPY simulator

Display area for ETM trace

The Trace window contains the following columns for ETM trace:

Trace window column	Description
Index	A number that corresponds to each ETM packet. Examples of ETM packets are instructions, synchronization points, and exception markers.
Frame Time	<p>When collecting trace data in cycle-accurate mode (requires ARM7/9)—enable Cycle accurate tracing in the ETM Trace Settings dialog box—the value corresponds to the number of elapsed cycles since the start of the execution. This column is only available for the J-Link/J-Trace driver.</p> <p>When collecting trace data in non-cycle-accurate mode, the value corresponds to an approximate amount of cycles. For Cortex-M devices, the value is repeatedly calibrated with the actual number of cycles.</p> <p>When the Show timestamp option is selected in the ETM Trace Settings dialog box, the value displays the time instead of cycles. To display the value as time requires collecting data in cycle-accurate mode, see <i>Cycle accurate tracing</i>, page 9, and the J-Link/J-Trace driver.</p>
Address	The address of the executed instruction.
Opcode	The operation code of the executed instruction.
Trace	The collected sequence of executed machine instructions. Optionally, the corresponding source code can also be displayed.
Comment	This column is only available for the J-Link/J-Trace driver.

Table 3: Trace window columns for ETM trace

Note: For RDI drivers, this window looks slightly different.

Display area for SWO trace

The SWO Trace window displays all data from the SWO communication channel in these columns:

Trace window column	Description
Index	An index number for each row in the trace buffer. Simplifies the navigation within the buffer.
SWO Packet	The contents of the captured SWO packet.

Table 4: Trace window columns for SWO trace

Trace window column	Description
Cycles	The approximate number of cycles from the start of the execution until the event.
Event	The event type of the captured SWO packet. If the column displays <code>Overflow</code> , the data packet could not be sent, because too many SWO features use the SWO channel at the same time. To decrease the amount of transmissions on the communication channel, point at the SWO button—on the IDE main window toolbar—with the mouse pointer to get detailed tooltip information about which C-SPY features that have requested trace data generation. Disable some of the features.
Value	The event value, if any.
Trace	If the event is a sampled PC value, the instruction is displayed in the Trace column. Optionally, the corresponding source code can also be displayed.
Comment	Additional information. This includes the values of the selected Trace Events counters, or the number of the comparator (hardware breakpoint) used for the Data Log breakpoint.

Table 4: Trace window columns for SWO trace (Continued)



If the display area seems to show garbage, make sure you specified correct value for the **CPU clock** option in the **SWO Configuration** dialog box.

TRACE SAVE DIALOG BOX

Use the **Trace Save** dialog box—available from the driver-specific menu, and from the Trace window and the SWO Trace window—to save the collected trace data, as it is displayed in the window, to a file.

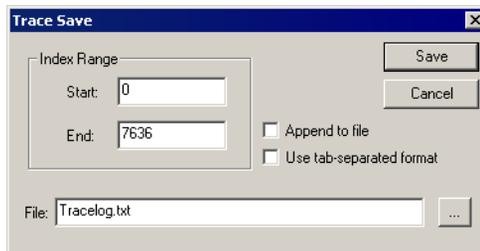


Figure 5: Trace Save dialog box

Index Range

Use this option to save a range of frames to a file. Specify a start index and an end index (as numbered in the index column in the Trace window).

Append to file

Appends the trace data to an existing file.

Use tab-separated format

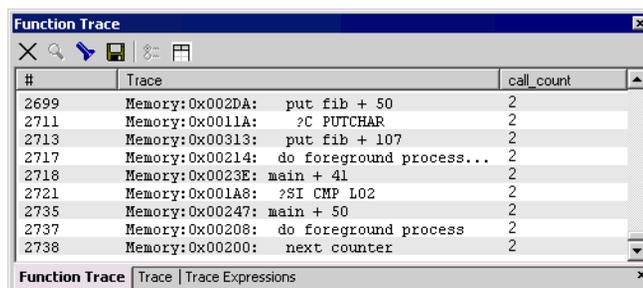
Saves the content in columns that are tab separated, instead of separated by white spaces.

File

Use this text box to locate a file for the trace data.

FUNCTION TRACE WINDOW

The Function Trace window is available from the driver-specific menu when you are using the C-SPY simulator or any driver that support ETM trace.



#	Trace	call_count
2699	Memory:0x002DA: put fib + 50	2
2711	Memory:0x0011A: ?C PUTCHAR	2
2713	Memory:0x00313: put fib + 107	2
2717	Memory:0x00214: do foreground process...	2
2718	Memory:0x0023E: main + 41	2
2721	Memory:0x001A8: ?SI CMP L02	2
2735	Memory:0x00247: main + 50	2
2737	Memory:0x00208: do foreground process	2
2738	Memory:0x00200: next counter	2

Figure 6: Function Trace window

Using this window

The Function Trace window displays a subset of the trace data displayed in the Trace window. Instead of displaying all rows, the Function Trace window only shows trace data corresponding to calls to and returns from functions.

Toolbar

For information about the toolbar, see *Trace toolbar*, page 16.

The display area

For information about the columns in the display area, see:

- *Trace display area in the C-SPY simulator*, page 17
- *Display area for ETM trace*, page 18.

TIMELINE WINDOW

The Timeline window is available from the **Simulator** menu and from the **J-Link** menu during a debug session.

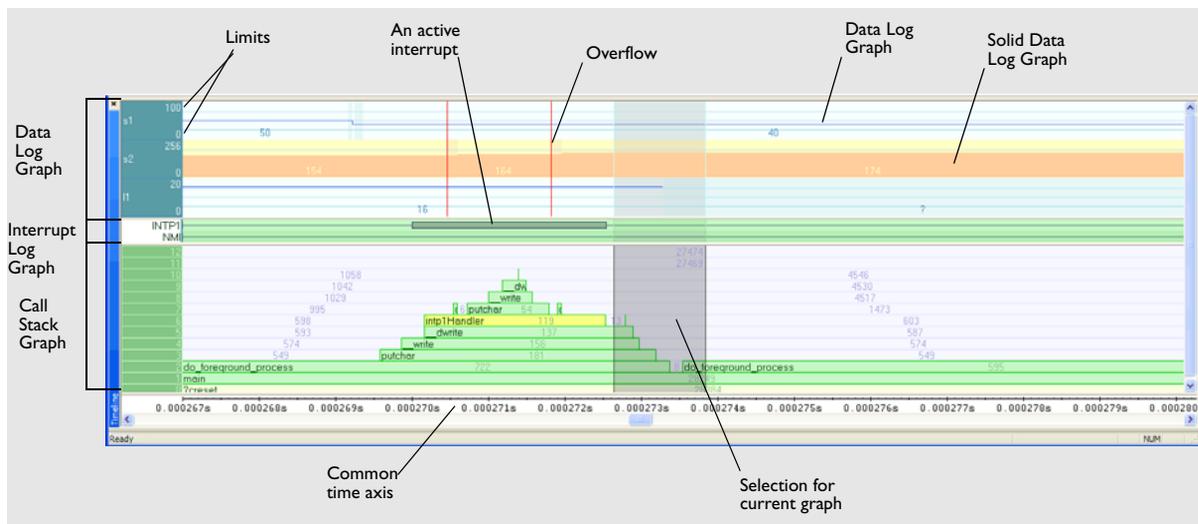


Figure 7: Timeline window

Using this window

The Timeline window displays trace data—for interrupt logs, data logs, and for the call stack—as graphs in relation to a common time axis.

Display area

The display area can be populated with up to four different graphs:

- Interrupt Log Graph
- Data Log Graph
- Call Stack Graph

- Power Log Graph.

At the bottom of the window, there is a common time axis that uses seconds or cycles as the time unit.

Interrupt Log Graph

The Interrupt Log Graph displays interrupts reported by SWO trace or by the C-SPY simulator. In other words, the graph provides a graphical view of the interrupt events during the execution of your application, where:

- The label area at the left end of the graph shows the names of the interrupts.
- The graph itself shows active interrupts as a thick green horizontal bar. This graph is a graphical representation of the information in the Interrupt Log window, see *Interrupt Log window* (in *IAR Embedded Workbench® IDE User Guide for ARM®*).

Data Log Graph

The Data Log Graph displays the data logs generated by SWO trace or by the C-SPY simulator, for up to four different variables or address ranges specified as Data Log breakpoints, where:

- Each graph is labeled with—in the left-side area—the name or address for which you have specified the Data Log breakpoint.
- The graph itself displays how the value of the variable changes over time. The label area also displays the limits, or range, of the y-axis for a variable. You can use the context menu to change these limits. The graph can be displayed either as a thin line or as a color-filled solid graph. The graph is a graphical representation of the information in the Data Log window, see *Data Log window* (in *IAR Embedded Workbench® IDE User Guide for ARM®*).
- A red vertical line indicates overflow, which means that the communication channel failed to transmit all data logs from the target system.

Call Stack Graph

The Call Stack Graph displays the sequence of calls and returns collected by ETM trace. At the bottom of the graph you will usually find `main`, and above it, the functions called from `main`, and so on. The horizontal bars, which represent invocations of functions, use four different colors:

- Medium green for normal C functions with debug information
- Light green for functions known to the debugger only through an assembler label
- Medium or light yellow for interrupt handlers, with the same distinctions as for green.

The numbers represent the number of cycles spent in, or between, the function invocations.

Power Log Graph

The Power Log Graph displays power measurement samples generated by the debug probe or associated hardware.

Selection and navigation

Click and drag to select. The selection extends vertically over all graphs, but appears highlighted in a darker color for the selected graph. You can navigate backward and forward in the selected graph using the left and right arrow keys. Use the Home and End keys to move to the first or last relevant point, respectively. Use the navigation keys in combination with the Shift key to extend the selection.

Context menu

The context menu in the Timeline window contains some commands that are common to all graphs and some commands that are specific to each graph.



Figure 8: Timeline window context menu for Power Log Graph

Note: This is the context menu for the Power Log Graph, which means that the menu looks slightly different for the Data Log Graph, Interrupt Log Graph, and for the Call Stack Graph.

These commands are available on the context menu:

Menu command	Applies to	Description
Navigate	All graphs	Commands for navigating over the graph(s); choose between: Next moves the selection to the next relevant point in the graph. Shortcut key: right arrow. Previous moves the selection backward to the previous relevant point in the graph. Shortcut key: left arrow. First moves the selection to the first data entry in the graph. Shortcut key: Home. Last moves the selection to the last data entry in the graph. Shortcut key: End. End moves selection to the last data in any displayed graph, in other words the end of the time axis. Shortcut key: Ctrl+End.
Auto Scroll	All graphs	Toggles auto scrolling on or off. When on, the most recent collected data is automatically displayed.
Zoom	All graphs	Commands for zooming the window, in other words, changing the time scale; choose between: Zoom to Selection makes the current selection fit the window. Shortcut key: Return. Zoom In enlarges the time scale. Shortcut key: +. Zoom Out shrinks the time scale. Shortcut key: -. 10ns, 100ns, 1us , etc makes an interval of 10 nanoseconds, 100 nanoseconds, 1 microseconds, respectively, fit the window. 1ms, 10ms , etc makes an interval of 1 milliseconds or 10 milliseconds, respectively, fit the window. 10m, 1h , etc makes an interval of 10 minutes or 1 hour, respectively, fit the window.
Data Log	Data Log Graph	A heading that shows that the Data Log-specific commands below it are available.
Power Log	Power Log Graph	A heading that shows that the Power Log-specific commands below it are available.
Call Stack	Call Stack Graph	A heading that shows that the Call Stack Log-specific commands below it are available.
Interrupts	Interrupt Log Graph	A label that shows that the Interrupt Log-specific commands below it are available.

Table 5: Commands on the Timeline window context menu

Menu command	Applies to	Description
Enable	All graphs	Toggles the display of the graph. If you disable a graph, that graph will be indicated as OFF in the Timeline window. If no trace data has been collected for a graph, <i>no data</i> will appear instead of the graph.
Variable	Data Log Graph	The name of the variable for which the following Data Log-specific commands apply. This menu entry is context-sensitive, which means it reflects the Data Log Graph you selected in the Timeline window (one of up to four).
Solid Graph	Data Log Graph	Displays the graph as a color-filled solid graph instead of as a thin line.
Viewing range	Data Log Graph Power Log Graph	Displays the Viewing Range dialog box, see <i>Viewing Range dialog box</i> , page 28.
Size	Data Log Graph Power Log Graph	Determines the vertical size of the graph; choose between Small , Medium , and Large .
Show Numerical Values	Data Log Graph Power Log Graph	Shows the numerical value of the variable, in addition to the graph.
Go To Source	Common	Displays the corresponding source code in an editor window, if applicable.
Select Graphs	Common	Selects which graphs are to be displayed in the Timeline window.
Time Axis Unit	Common	Selects the unit used in the time axis; choose between Seconds and Cycles .
Profile Selection	Common	Enables profiling time intervals in the Function Profiler window.

Table 5: Commands on the Timeline window context menu (Continued)

POWER LOG WINDOW

The Power Log window is available from the **Simulator** menu and from the **J-Link** menu during a debug session.

Time	Program Counter	Current [mA]
<i>910.167us</i>	0x0800A404	128
<i>12969.875us</i>	0x08009654	103
<i>30490.563us</i>	0x0800A37E	122
<i>45053.250us</i>	0x0800A34C	122
<i>61436.250us</i>	0x0800A36C	122
<i>78274.333us</i>	0x0800A34E	122
<i>93064.542us</i>	0x0800A360	135
<i>107399.667us</i>	0x0800A358	122
<i>129016.125us</i>	0x0800A356	122

Figure 9: Power Log window

Using this window

The Power Log window displays generated power samples.

Display area

The Power Log window contains these columns:

Column	Description
Time	The time for the data access, based on the clock frequency specified in the SWO Configuration dialog box. If the time is displayed in italics, the target system has not been able to collect a correct time, but instead had to approximate it. This column is available when you have selected Show time from the context menu.
Cycles	The number of cycles from the start of the execution until the event. This information is cleared at reset. If a cycle is displayed in italics, the target system has not been able to collect a correct time, but instead had to approximate it. This column is available when you have selected Show cycles from the context menu.
Program Counter	The content of the PC, that is, the address of the instruction that performed the memory access. If the column displays ---, the target system failed to provide the debugger with any information. If the column displays <i>Overflow</i> in red, the communication channel failed to transmit all data from the target system.

Table 6: Power Log window columns

Column	Description
Current [mA]	The power measurement value, expressed in milliamperes. The grey area graphically visualizes the power value in percentage of the Factory setting received from the properties of the measuring hardware; see <i>Viewing Range dialog box</i> , page 28.

Table 6: Power Log window columns (Continued)

Context menu

The Power Log window contains this context menu:

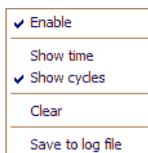


Figure 10: Power Log window context menu

Note: The commands are the same in each window, but they only operate on the specific window.

These commands are available on the menu:

Menu command	Description
Enable	Enables the logging system. The system will log information also when the window is closed.
Show time	Displays the Time column in the Power Log window, Data Log window, and in the Interrupt Log window, respectively.
Show cycles	Displays the Cycles column in the Data Log window and in the Interrupt Log window, respectively.
Clear	Deletes the log information. Note that this will happen also when you reset the debugger, or if you change the execution frequency in the SWO Setup dialog box.
Save to log file	Displays a dialog box where you can select the destination file for the log information. The entries in the log file are separated by TAB and LF. An X in the Approx column indicates that the time stamp is an approximation.

Table 7: Commands on the Power Log window context menu

VIEWING RANGE DIALOG BOX

The **Viewing Range** dialog box is available from the context menu that appears when you right-click in the Power Log window or the Data Log window.

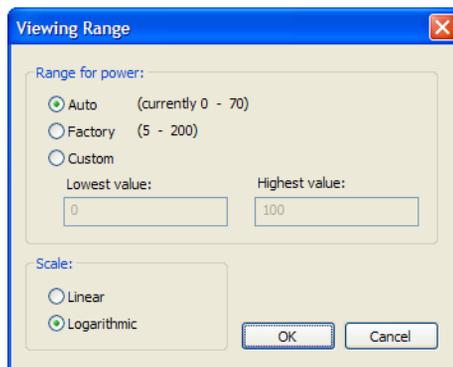


Figure 11: Viewing Range dialog box

This dialog box is available for the Data Log Graph and for the Power Log Graph.

Use this dialog box to specify the value range, that is, the range for the Y-axis for the graph.

Range for xxxx

Selects the viewing range for the displayed values:

Auto	Uses the range according to the range of the values that are actually collected, continuously keeping track of minimum or maximum values. The currently computed range, if any, is displayed in parentheses. The range is rounded to reasonably <i>even</i> limits.
Factory	For the Data Log Graph: Uses the range according to the value range of the variable, for example 0–65535 for an unsigned 16-bit integer. For the Power Log Graph: Uses the range according to the properties of the measuring hardware.
Custom	Use the text boxes to specify an explicit range.

Scale

Selects the scale type of the Y-axis::

- **Linear**
- **Logarithmic.**

TRACE START BREAKPOINTS DIALOG BOX

The **Trace Start** dialog box is available from the context menu that appears when you right-click in the Breakpoints window.

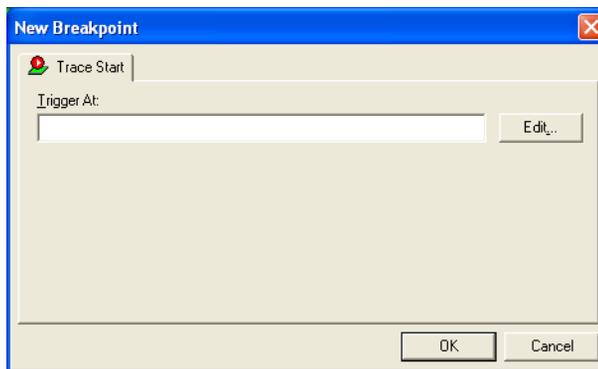


Figure 12: Trace Start breakpoints dialog box

Using this dialog box

This dialog box is available for the C-SPY simulator. See also *Using J-Link trace triggers and trace filters in IAR Embedded Workbench® IDE User Guide for ARM®*.

To set a Trace Start breakpoint:

- 1 In the editor or Disassembly window, right-click and choose **Trace Start** from the context menu.
- 2 Alternatively, open the Breakpoints window by choosing **View>Breakpoints**.
- 3 In the Breakpoints window, right-click and choose **New Breakpoint>Trace Start**.
Alternatively, to modify an existing breakpoint, select a breakpoint in the Breakpoints window and choose **Edit** on the context menu.
- 4 In the **Trigger At** text box specify an expression, an absolute address, or a source location. Click **OK**.
- 5 When the breakpoint is triggered, the trace data collection starts.

Trigger At

Specify the location for the breakpoint in the **Trigger At** text box. Alternatively, click the **Edit** browse button to open the **Enter Location** dialog box; see *Enter Location dialog box (UARM17b)*.

TRACE STOP BREAKPOINTS DIALOG BOX

The **Trace Stop** dialog box is available from the context menu that appears when you right-click in the Breakpoints window.

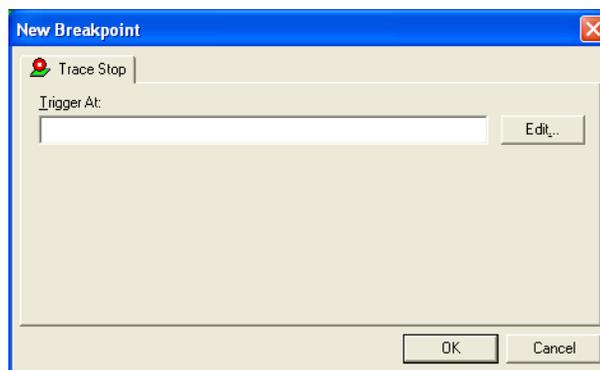


Figure 13: Trace Stop breakpoints dialog box

Using this dialog box

This dialog box is available for the C-SPY simulator. See also *Using J-Link trace triggers and trace filters* (in *IAR Embedded Workbench® IDE User Guide for ARM®*).

To set a Trace Stop breakpoint:

- 1 In the editor or Disassembly window, right-click and choose **Trace Stop** from the context menu.
- 2 Alternatively, open the Breakpoints window by choosing **View>Breakpoints**.
- 3 In the Breakpoints window, right-click and choose **New Breakpoint>Trace Stop**.

Alternatively, to modify an existing breakpoint, select a breakpoint in the Breakpoints window and choose **Edit** on the context menu.

- 4 In the **Trigger At** text box specify an expression, an absolute address, or a source location. Click **OK**.
- 5 When the breakpoint is triggered, the trace data collection stops.

Trigger At

Specify the location for the breakpoint in the **Trigger At** text box. Alternatively, click the **Edit** browse button to open the **Enter Location** dialog box.

TRACE EXPRESSIONS WINDOW

The Trace Expressions window is available from the Trace window toolbar when you are using the C-SPY simulator.

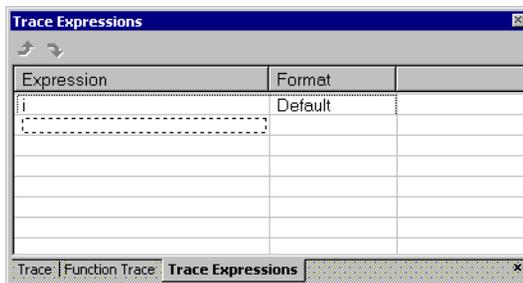


Figure 14: Trace Expressions window

Using this window

In the Trace Expressions window you can specify, for example, a specific variable (or an expression) for which you want to collect trace data.

Toolbar

Use the toolbar buttons to change the order between the expressions:

Toolbar button	Description
Arrow up	Moves the selected row up.
Arrow down	Moves the selected row down.

Table 8: Toolbar buttons in the Trace Expressions window

Display area

In the display area you can specify expressions for which you want to collect trace data:

Column	Description
Expression	Specify any expression that you want to collect data from. You can specify any expression that can be evaluated, such as variables and registers.
Format	Shows which display format that is used for each expression. Note that you can change display format via the context menu.

Table 9: Trace Expressions window columns

Each row in this window will appear as an extra column in the Trace window.

FIND IN TRACE DIALOG BOX

The **Find in Trace** dialog box is available by clicking the **Find** button on the Trace window toolbar or by choosing **Edit>Find and Replace>Find**.

Note that the **Edit>Find and Replace>Find** command is context-dependent. It displays the **Find in Trace** dialog box if the Trace window is the current window or the **Find** dialog box if the editor window is the current window.

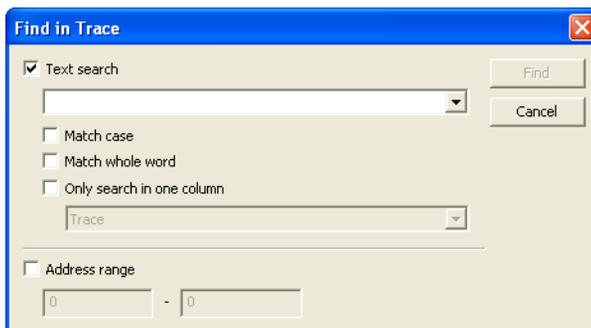


Figure 15: Find in Trace dialog box

Using this window

Use the **Find in Trace** dialog box to specify the search criteria for advanced searches in the trace data.

The search results are displayed in the Find in Trace window—available by choosing the **View>Messages** command, see *Find in Trace window*, page 33.

See also, *Searching in trace data*, page 6.

You specify the search criteria with the following options.

Text search

A text field where you type the string you want to search for. Use these options to fine-tune the search:

- Match Case** Searches only for occurrences that exactly match the case of the specified text. Otherwise specifying `int` will also find `INT` and `Int`.
- Match whole word** Searches only for the string when it occurs as a separate word. Otherwise `int` will also find `print`, `sprintf` and so on.
- Only search in one column** Searches only in the column you selected from the drop-down list.

Address Range

Use the text fields to specify an address range. The trace data within the address range is displayed. If you also have specified a text string in the **Text search** field, the text string is searched for within the address range.

FIND IN TRACE WINDOW

The Find in Trace window is available from the **View>Messages** menu. Alternatively, it is automatically displayed when you perform a search using the **Find in Trace** dialog box.

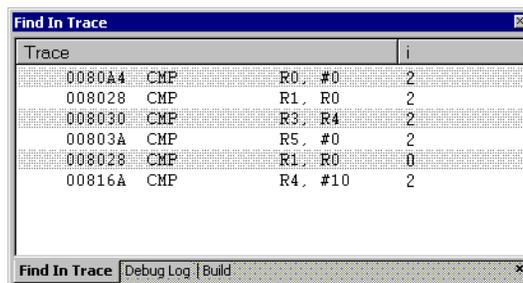


Figure 16: Find in Trace window

Using this window

The Find in Trace window displays the result of searches in the trace data. Double-click an item in the Find in Trace window to bring up the same item in the Trace window.

Before you can view any trace data, you must specify the search criteria in the **Find in Trace** dialog box, see *Find in Trace dialog box*, page 32.

See also, *Searching in trace data*, page 6.

Display area

The Find in Trace window looks like the Trace window and shows the same columns and data, but *only* those rows that match the specified search criteria.

Using the profiler

This chapter gives you information about using the profiler. More specifically, these topics are covered:

- *Introduction to the profiler*, page 35
- *Procedures for using the profiler*, page 37
- *Related reference information*, page 39.

Introduction to the profiler

This section gives you information about:

- *Reasons for using the profiler*, page 35
- *Briefly about the profiler*, page 35
- *Requirements for using the profiler*, page 36.

REASONS FOR USING THE PROFILER

Function profiling can help you find the functions where most time is spent during execution. Focus on those functions when optimizing your code. A simple method of optimizing a function is to compile it using speed optimization. Alternatively, you can move the data used by the function into the memory which uses the most efficient addressing mode. For detailed information about efficient memory usage, see the *IAR C/C++ Development Guide for ARM®*.

Instruction profiling can help you fine-tune your code on a very detailed level, especially for assembler source code. Instruction profiling can also help you to understand where your compiled C/C++ source code spends most of its time, and perhaps give insight into how to rewrite it for better performance.

BRIEFLY ABOUT THE PROFILER

Function profiling information is displayed in the Function Profiler window, that is, timing information for the functions in an application. Profiling must be turned on explicitly using a button on the window's toolbar, and will remain enabled until it is turned off.

Instruction profiling information is displayed in the Disassembly window, that is, the number of times each instruction has been executed.

Profiling sources

The profiler can use different mechanisms, or *sources*, to collect profiling information. Depending on the available hardware features, one or more of the sources can be used for profiling:

- Trace (calls)

The full instruction trace (ETM trace) is analyzed to determine all function calls and returns. When the collected instruction sequence is incomplete or discontinuous, as sometimes happens when using ETM trace, the profiling information is less accurate.
- Trace (flat) / Sampling

Each instruction in the full instruction trace (ETM trace) or each PC Sample (from SWO trace) is assigned to a corresponding function or code fragment, without regard to function calls or returns. This is most useful when the application does not exhibit normal call/return sequences, such as when you are using an RTOS, or when you are profiling code which does not have full debug information.
- Breakpoints

The profiler sets a breakpoint on every function entry point. During execution, the profiler collects information about function calls and returns as each breakpoint is hit. This assumes that the hardware supports a large number of breakpoints, and it has a huge impact on execution performance.

Power Sampling

Some debug probes support regular sampling of the power consumption of the development board, or components on the board. Each sample is also associated with a PC sample and represents the power consumption (actually, the electrical current) for a small time interval preceding the time of the sample. When the profiler is set to use *Power Sampling*, additional columns are displayed in the Profiler window. Each power sample is associated with a function or code fragment, just as with regular PC Sampling. Note that this does not imply that all the energy corresponding to a sample can be attributed to that function or code fragment. The time scales of power samples and instruction execution are vastly different; during one power measurement, the CPU has typically executed many thousands of instructions. Power Sampling is a statistics tool.

REQUIREMENTS FOR USING THE PROFILER

To use the profiler, you need one of these setups:

- A J-Link or a J-Trace debug probe with an SWD/SWO interface between the probe and the target system, which must be based on a Cortex-M device
- A J-Trace debug probe and an ARM7/9 device with ETM trace.
- A J-Link or J-Trace Ultra probe.

Procedures for using the profiler

This section gives you information about:

- *How to use the profiler on function level*, page 37
- *How to use the profiler on instruction level*, page 37
- *Selecting a time interval for profiling information*, page 38.

HOW TO USE THE PROFILER ON FUNCTION LEVEL

To display function profiling information in the Function Profiler window, follow these steps:

- 1 Make sure you build your application using these options:

Category	Setting
C/C++ Compiler	Output>Generate debug information
Linker	Output>Include debug information in output

Table 10: Project options for enabling the profiler

- 2 To set up the profiler for function profiling:
 - If you use ETM trace, make sure that the **Cycle accurate tracing** option is selected in the **Trace Settings** dialog box.
 - If you use the SWD/SWO interface, no specific settings are required.
- 3  When you have built your application and started C-SPY, choose **J-Link>Function Profiler** to open the Function Profiler window, and click the **Enable** button to turn on the profiler. Alternatively, choose **Enable** from the context menu that is available when you right-click in the Function Profiler window.
- 4 Start executing your application to collect the profiling information.
- 5 Profiling information is displayed in the Function Profiler window. To sort, click on the relevant column header.
- 6  When you start a new sampling, you can click the **Clear** button—alternatively, use the context menu—to clear the data.

HOW TO USE THE PROFILER ON INSTRUCTION LEVEL

To display instruction profiling information in the Disassembly window, follow these steps:

- 1 When you have built your application and started C-SPY, choose **View>Disassembly** to open the Disassembly window, and choose **Instruction Profiling>Enable** from the context menu.

- 2 Make sure that the **Show** command on the context menu is selected, to display the profiling information.
- 3 Start executing your application program to collect the profiling information.
- 4 When the execution stops, for instance because the program exit is reached or a breakpoint is triggered, you can view instruction level profiling information in the left-hand margin of the Disassembly window.

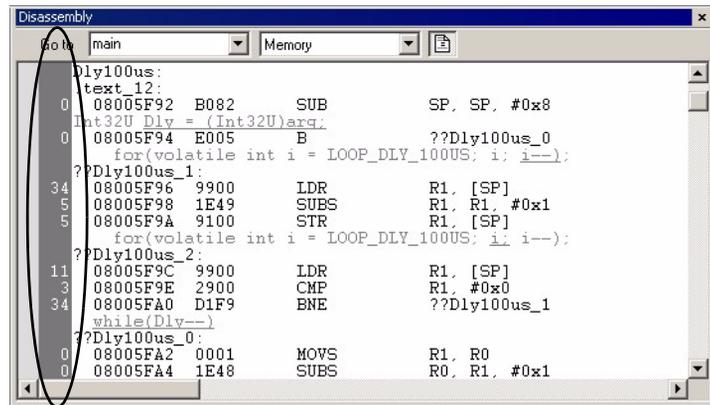


Figure 17: Instruction count in Disassembly window

For each instruction, the number of times it has been executed is displayed.

If more than one source for the profiling data is supported by the C-SPY driver that you are using, the driver will try to use trace information as the source, unless you have chosen to use a different source. You can change the source to be used from the context menu that is available in the Profiler window.

SELECTING A TIME INTERVAL FOR PROFILING INFORMATION

Normally, the profiler computes its information from all PC samples it receives, accumulating more and more information until you explicitly clear the profiling information. However, you can choose a time interval for which the profiler computes the PC samples. This function is supported by the J-Link/J-Trace Ultra probe.

To select a time interval, follow these steps:

- 1 Choose **Function Profiler** from the driver-specific menu.
- 2 In the Function Profiler window, right-click and choose **Source: Sampling** from the context menu.

- 3 Execute your application to collect samples.
- 4 Choose **View>Timeline**.
- 5 In the Timeline window, right-click and drag to select a time interval.

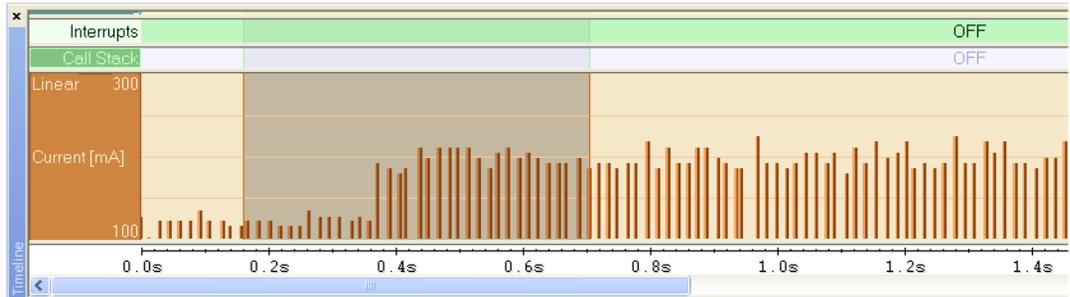


Figure 18: Timeline window with a selected time interval—a selection

- 6 In the selected time interval, right-click and choose **Profile Selection** from the context menu.
- 7 The Function Profiler window now displays profiling information for the selected time interval.

Function	PC Samples	PC Samples (%)	Power Samples	Energy (%)	Avg
GetButtons()	791	33.10	9	30.82	19
Dly100us(void *)	463	19.37	7	15.38	12
GLCD_SPI_TranserByte(Int3...	353	14.77	4	8.32	12
memcmp	325	13.60	4	14.64	21
main()	288	12.05	6	20.07	19
GLCD_Backlight(Int8U)	108	4.52	2	6.77	19
GLCD_SendCmd(GLCD_Cm...	43	1.80	0	0.00	-
GLCD_SPI_SendBlock(plnt8...	19	0.79	2	4.00	11
GLCD_SetWindow(Int32U, Int...	0	0.00	0	0.00	-
GLCD_SetReset(Boolean)	0	0.00	0	0.00	-

Figure 19: Function Profiler window in time-interval mode

- 8 Click the **Full/Time-interval profiling** button to toggle the Full profiling view.

Related reference information

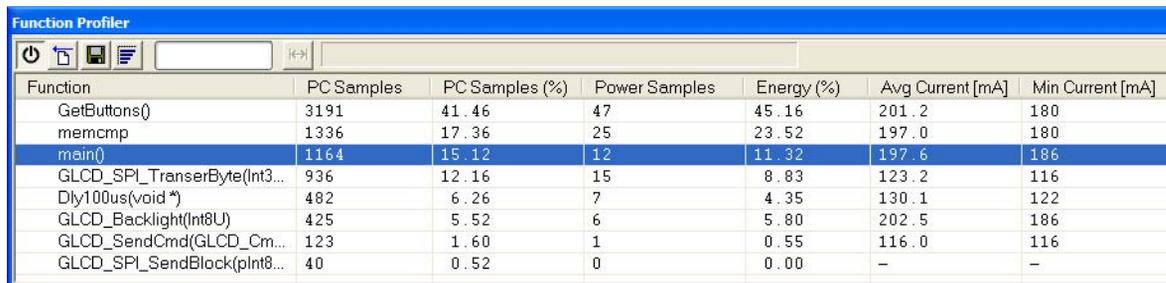
To use the profiler system, you might need reference information about these windows:

- *Function Profiler window*, page 40

- Disassembly window
- *SWO Configuration dialog box*, page 12.

FUNCTION PROFILER WINDOW

The Function Profiler window—available from the **J-Link** menu or the **Simulator** menu—displays function profiling information.



Function	PC Samples	PC Samples (%)	Power Samples	Energy (%)	Avg Current [mA]	Min Current [mA]
GetButtons()	3191	41.46	47	45.16	201.2	180
memcpy	1336	17.36	25	23.52	197.0	180
main()	1164	15.12	12	11.32	197.6	186
GLCD_SPL_TranserByte(Int3...	936	12.16	15	8.83	123.2	116
Dly100us(void*)	482	6.26	7	4.35	130.1	122
GLCD_Backlight(Int8U)	425	5.52	6	5.80	202.5	186
GLCD_SendCmd(GLCD_Cm...	123	1.60	1	0.55	116.0	116
GLCD_SPL_SendBlock(plnt8...	40	0.52	0	0.00	-	-

Figure 20: Function Profiler window based on the source Sampling

Toolbar

The toolbar at the top of the window provides these buttons:

Toolbar button	Description
	Enable/Disable Enables or disables the profiler.
	Clear Clears all profiling data.
	Graphical view Overlays the values in the percentage columns with a graphical bar.
	Save Opens a standard Save As dialog box where you can save the contents of the window, with tab-separated columns. Only non-expanded rows are included in the list file.

Table 11: Function Profiler window toolbar

Toolbar button	Description
<i>Progress bar</i>	Displays a backlog of profiling data that is still being processed. If the rate of incoming data is higher than the rate of the profiler processing the data, a backlog is accumulated. The progress bar indicates that the profiler is still processing data, but also approximately how far the profiler has come in the process. Note that because the profiler consumes data at a certain rate and the target system supplies data at another rate, the amount of data remaining to be processed can both increase and decrease. The progress bar can grow and shrink accordingly.
 Time-interval mode	Toggles between profiling a selected time interval or full profiling. This toolbar button is only available if PC Sampling is supported by the debug probe (SWO trace).
<i>Status field</i>	Displays the range of the selected time interval, in other words, the profiled selection. This field is yellow when Time-interval profiling mode is enabled. This field is only available if PC Sampling is supported by the debug probe (SWO trace).

Table 11: Function Profiler window toolbar (Continued)

The display area

The content in the display area depends on which source is used for the profiling information:

- For the Breakpoints and Trace (calls) sources, the display area contains one line for each function compiled with debug information enabled. When some profiling information has been collected, it is possible to expand rows of functions that have called other functions. The child items for a given function list all the functions that have been called by the parent function and the corresponding statistics.
- For the Sampling and Trace (flat) sources, the display area contains one line for each C function of your application, but also lines for sections of code from the runtime library or from other code without debug information, denoted only by the corresponding assembler labels. This is because it is only known for each sample where the application was executing at that specific point in time. If the PC is sampled inside a runtime library function, it is not possible to know from which C function it was called.

More specifically, the display area provides this information:

Column	Applies to source	Description
Function	All sources	The name of the profiled C function. For Sampling source, also sections of code from the runtime library or from other code without debug information, denoted only by the corresponding assembler labels, is displayed.
Calls	Breakpoints and Builtin	The number of times the function has been called.
Flat time	Breakpoints and Builtin	The time in cycles spent inside the function.
Flat time (%)	Breakpoints and Builtin	Flat time in cycles expressed as a percentage of the total time.
Acc. time	Breakpoints and Builtin	The time in cycles spent in this function and everything called by this function.
Acc. time (%)	Breakpoints and Builtin	Accumulated time in cycles expressed as a percentage of the total time.
PC Samples	Trace (flat) and Sampling	The number of PC samples associated with the function. For Sampling source, flat time shows the number of samples for a given function.
PC Samples (%)	Trace (flat) and Sampling	The number of PC samples associated with the function as a percentage of the total number of samples.
Power Samples	Power Sampling	The number of power samples associated with that row.
Energy (%)	Power Sampling	The accumulated value of all measurements associated with that row, expressed as a percentage of all measurements.
Avg Current [mA]	Power Sampling	The average measured value for all samples associated with that row.
Min Current [mA]	Power Sampling	The minimum measured value for all samples associated with that row.
Max Current [mA]	Power Sampling	The maximum measured value for all samples associated with that row.

Table 12: Function Profiler window columns

* For Sampling source, calls are not possible to deduce which means the column will be empty.

Function Profiler window context menu

This context menu is available in the Function Profiler window:

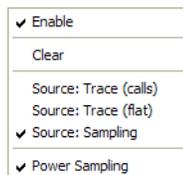


Figure 21: Function Profiler window context menu

These commands are available on the menu:

Menu command	Description
Enable	Enables the profiler. The system will collect information also when the window is closed.
Clear	Clears all profiling data.
Source*	Selects which source to be used for the profiling information. Choose between: Sampling , supported by the J-Link debug probe and the J-Trace debug probe. The instruction count for instruction profiling represents the number of samples for each instruction. Trace (calls) , supported by the J-Trace debug probe. The instruction count for instruction profiling is only as complete as the collected trace data. Trace (flat) , supported by the J-Trace debug probe. The instruction count for instruction profiling is only as complete as the collected trace data.
Power Sampling	Toggles power sampling information on or off. This command is supported by the J-Link and J-Trace Ultra debug probes.

Table 13: Commands on the Function Profiler window context menu

* Available sources depend on the C-SPY driver you are using.