

## Execution Coverage

### 1 Introduction

Execution coverage records all addresses being executed and as a result displays which code is executed or not executed. It can be used to verify a test code, which should exercise a complete target application. It also helps finding a so called “dead code”, the code that is never executed. Such code represents undesired overhead when assigning code memory resources.

Depending on the implementation, execution coverage operates either in off-line or real-time operation mode.

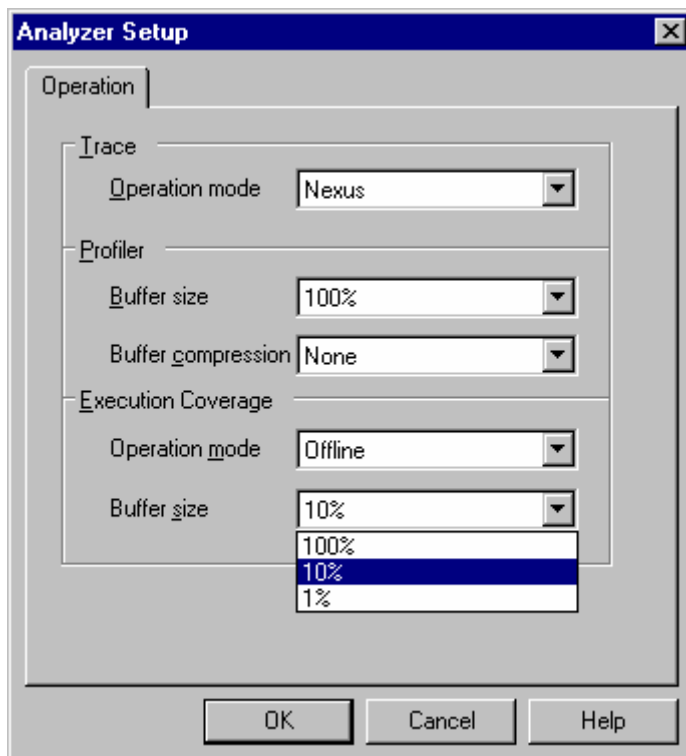
Off-line execution coverage features:

- decision coverage
- statement coverage, which should be used when real-time execution coverage is not available

Real-time execution coverage features:

- statement coverage running indefinitely

#### Off-line Execution Coverage



Initially, off-line execution coverage has been introduced on on-chip debug emulation systems with the message based trace port, where a concept used in the in-circuit emulation systems seemed to be not feasible. A standard trace is used to collect the program flow and then software analysis is performed off-line to obtain the execution coverage results. Time of program execution being tested directly depends on the target CPU speed and the trace buffer size. Upload time to the PC depends on the trace buffer size and the iC3000 unit. In case of iTRACE GT or Active GT development platform with 1GB or more trace buffer, it is recommended to use ic3000 GT unit which offers a high-speed trace upload to the PC. Off-line execution coverage trace buffer size is configurable in the 'Hardware/Analyzer Setup' dialog (1, 10 or 100%).

Off-line execution coverage analysis tests the following two metrics:

- **Statement Coverage** identifies which executable code was executed and which was not executed
- **Decision Coverage** exercises all conditional branch instructions and identifies if the instruction was taken, not taken, taken in both directions or not executed at all.

Off-line execution coverage has its advantages and disadvantages. It can be implemented on all development systems featuring program trace and features additional decision coverage metric comparing to the real-time execution coverage. Decision coverage is an add-on to the statement coverage (a standard feature on the in-circuit emulators). However, the execution time of the off-line execution coverage is limited by the trace buffer size and can never run infinitely. Further, due to the off-line trace based concept, it's impossible to identify which parts of the code were not tested or were tested but were not executed. Thereby, off-line execution coverage must be used with caution in order to obtain reliable and valid information for the code under the test.

A so called unit test, where the decision coverage becomes a necessity, is getting more and more important in the test departments. It is impossible to set up a test application and environment, which would systematically cover and test all possible paths in the target application. However, a unit test comes very close to this and can be run systematically. It first identifies all functions of the target application and then each function is individually stressed with different input parameters and the output results verified against expected ones. One of the output results is also decision coverage metric next to the statement coverage metric. Executed logical paths within the function are identified with the decision and statement coverage (while the function is exercised with input parameters).

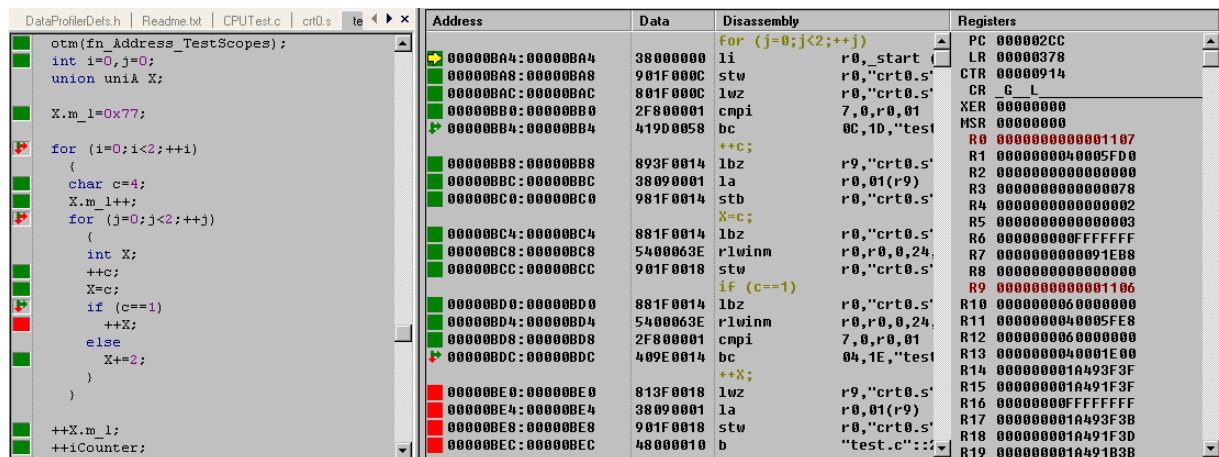
### Real-time Execution Coverage

Real-time execution coverage is based on a hardware logic, which in real-time monitors all executed program addresses. It features statement coverage but no decision coverage since it keeps the information on all executed program addresses but without any history, which would tell which address was executed when and in what order. A major advantage of the real-time execution coverage is that it can run **indefinitely**, which does not apply for the off-line coverage.

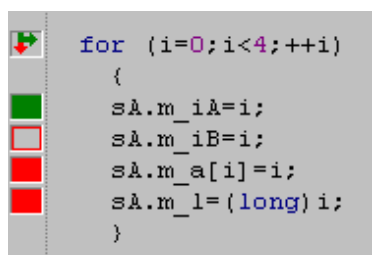
Real-time execution coverage is available for all in-circuit emulation systems with bus trace (including V850ES/Fx3 ActivePRO POD, which has different trace port type) and on-chip debug emulation systems with trace port, where iSYSTEM proprietary RTR technology is implemented (MPC55xx except for MPC551x, MPC56x, ARM ETM)

## 2 Results

Statement and decision coverage results are displayed in the execution coverage, the source and the disassembly window.



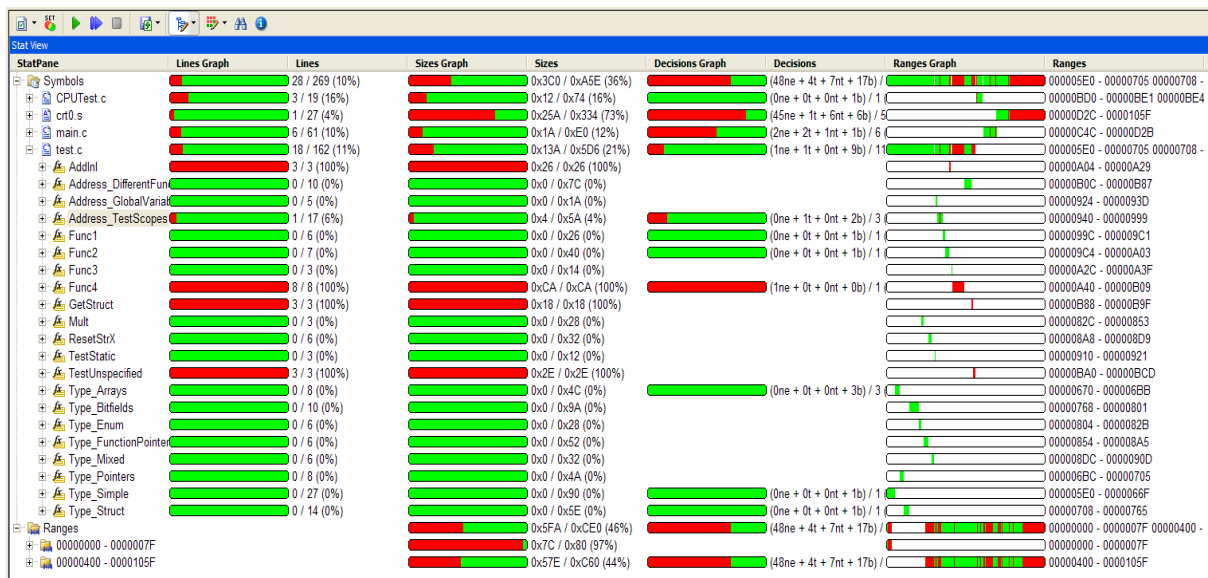
Source and disassembly window



Functions, which have not been executed, are marked with a red box. Fully executed functions are marked with a green box while partially executed functions are marked with a red frame.

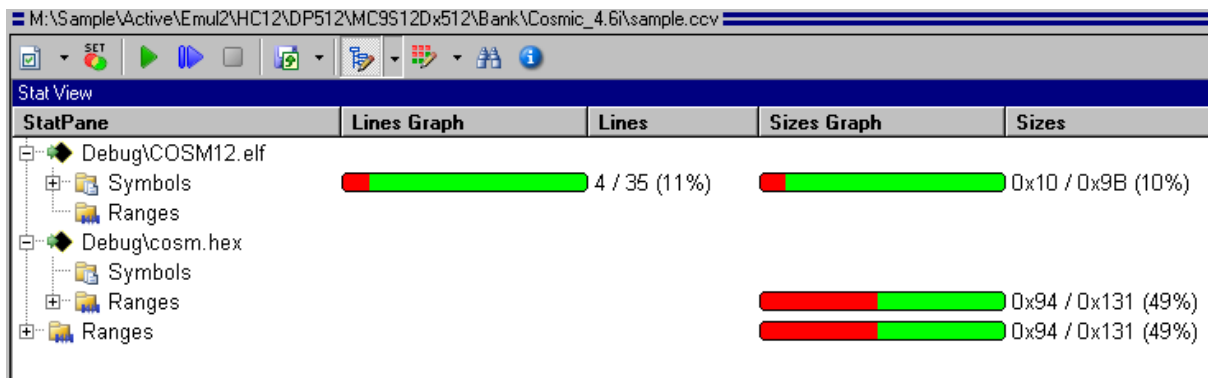
Red and green arrows in the source and disassembly window display decision coverage results. Looking in the disassembly window, the customer can identify a state of all conditional branches: taken, not taken, taken in both directions and not executed.

Decision coverage works on an object level and therefore the results are exact for the CPU instructions, which are displayed in the disassembly window. Decision coverage results displayed in the source window cannot be exact as the results in the disassembly window. Source lines consist of more instructions and therefore decision coverage information of more instructions is merged into a single source window information mark, which can no longer display the details available in the disassembly window. Decision column in the execution coverage window provides information on how many “branch” instructions are not executed (ne), taken in one direction (t), not taken (nt) or taken in both directions (b) within a source line.



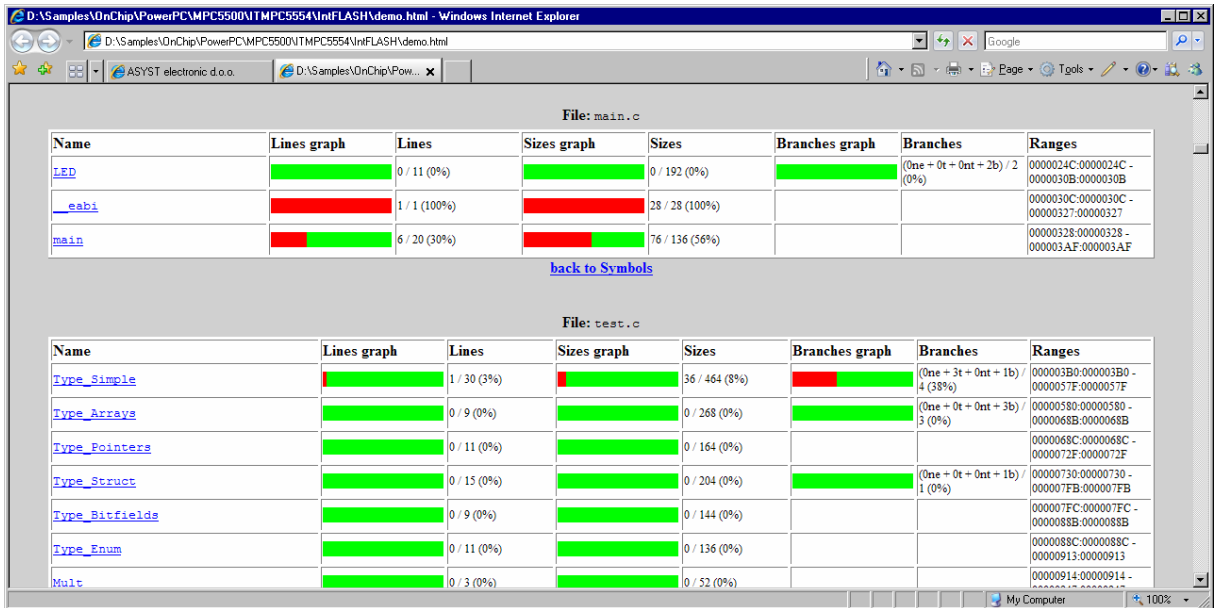
Execution coverage window

Note that in case of multiple download files, the execution coverage window displays results in respect to the symbols separately for every individual download file.



Execution Coverage results for project with two download files

Execution coverage results and projects settings can be saved in a file (.ccv) and can be open any time later. This allows analyzing and sharing execution coverage results without the emulator being hooked up. Additionally, execution coverage results can be saved in an html format for report and presentation purposes. Next chapter describes handling and use of execution coverage window.



HTML report

### 3 Toolbars



Toolbars are explained in order from the left to the right.

#### 3.1 Coverage configuration

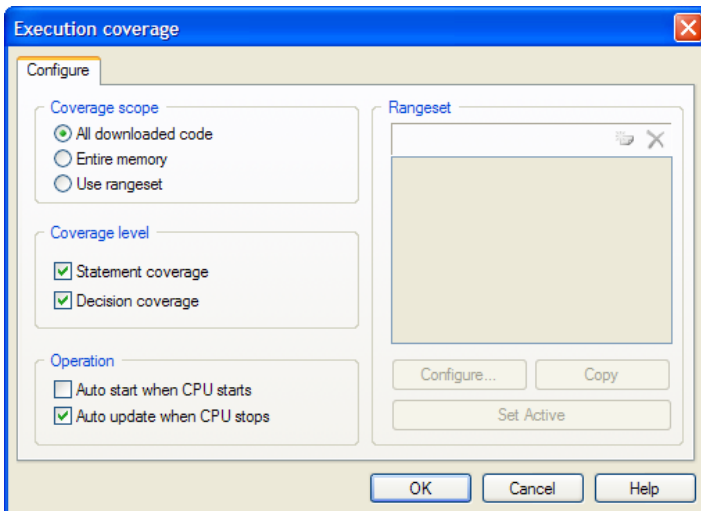
The subject of the execution coverage could be all downloaded code, entire memory or the particular program areas (ranges).

Display of Statement and Decision coverage can be individually enabled/disabled. Per default both are enabled.

Execution coverage can be started manually on request or automatically as soon as the program is run.

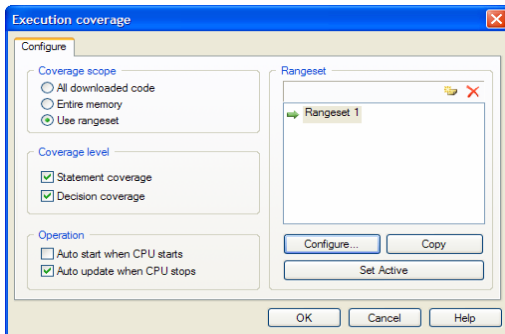
Results can be automatically updated every time the program is stopped or on request by stopping the execution coverage.

Note: Decision coverage is available for off-line execution coverage only

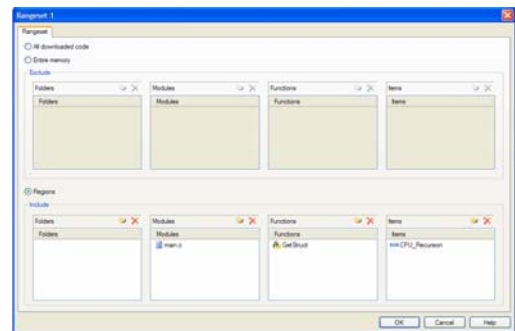


## 3.2 Configure rangeset

Program areas analyzed by the execution coverage can be configured through 'Coverage configuration' toolbar or 'Configure rangeset' toolbar or as a mixture of both. At the beginning, a symbol or a range is added and activated in the 'Coverage configuration' tab. This already defined range could be later quickly configured from the 'Configure rangeset' tab. Every newly covered area can be seen in the Symbols and Ranges tree structure in the execution coverage window.

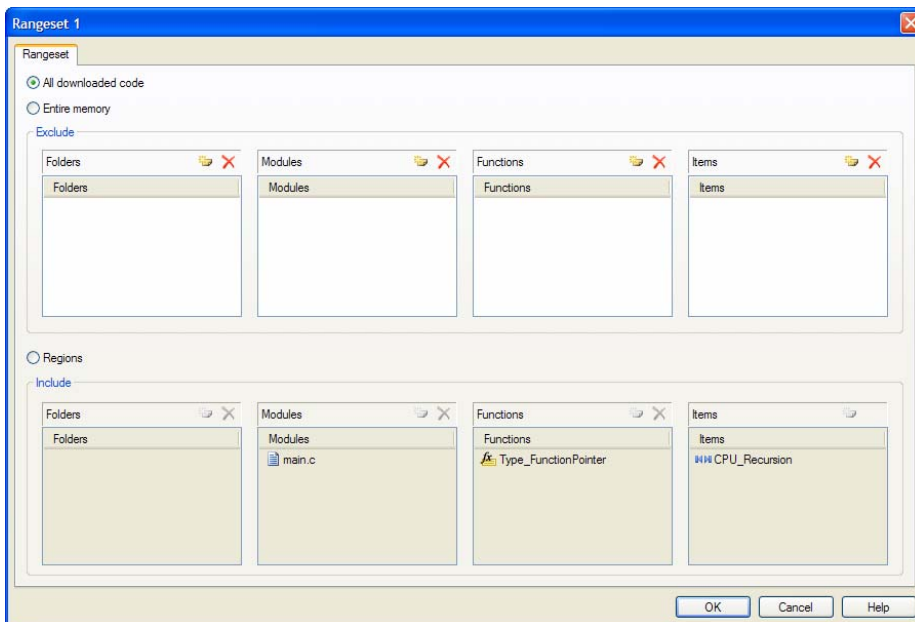


Rangeset definition/activation tab



Rangeset configuration tab

Available rangeset options can be seen on the below picture.



- Add all downloaded code

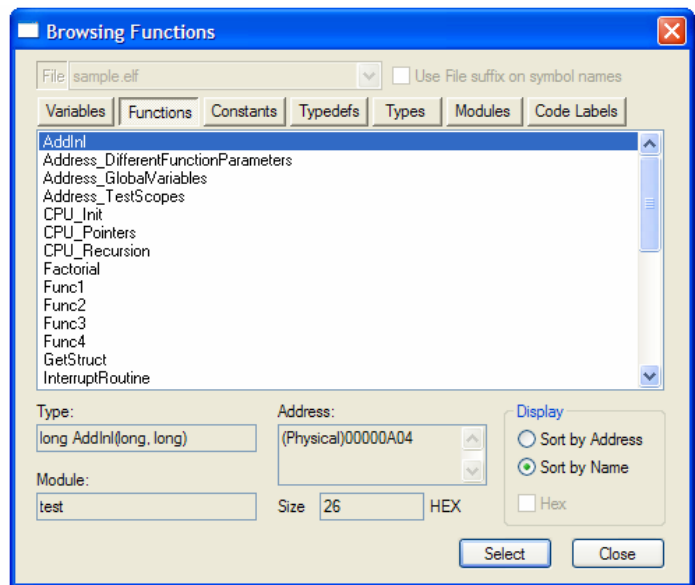
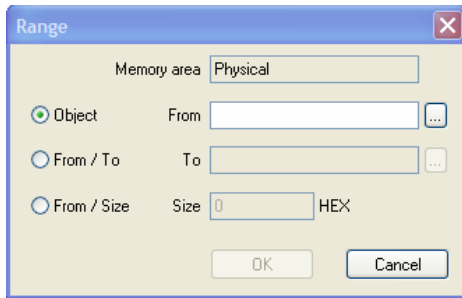
This configures execution coverage for all downloaded code. This is the most recommended configuration and works fine as long as there are no areas to be tested, which are not part of the download file and as long as the download file contains correct information. In case of any doubts, use other configuration choices.

- Add Entire memory

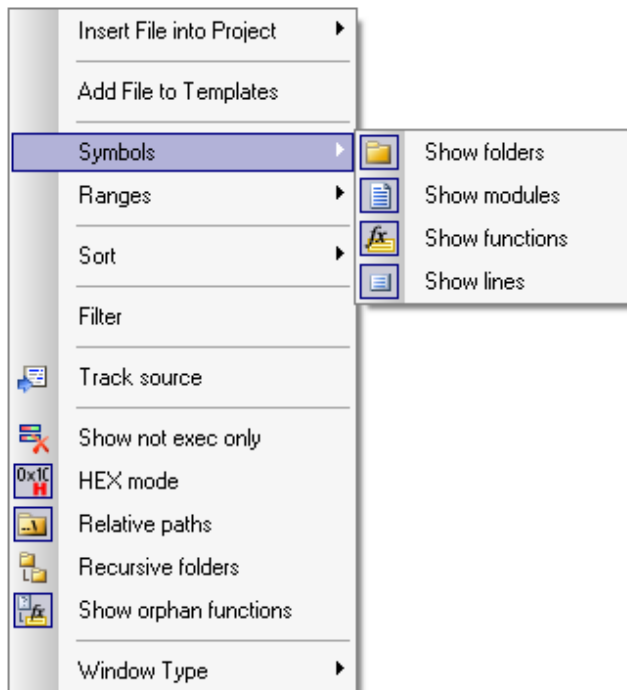
This configures execution coverage on the entire memory level with or without exclusions, which could be selected through particular folders, modules (files), functions and items. It yields a complete CPU address space for off-line execution coverage while for the real-time execution coverage working address space is limited by the hardware and covers first 8MBytes (MPC5500 Nexus RTR) of the CPU address space.

- Add Regions

Regions can be defined as an individual range item defined via address (as seen bellow on the left) or through the symbol table.



All other settings, like types of displayed symbols, ranges, sorting options, etc. are accessible through the right-click pop-up menu from the execution coverage main window.



### 3.3 *Begin*

Starts execution coverage session.

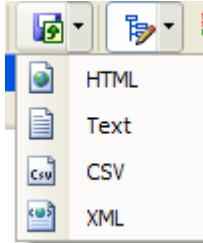
### 3.4 *Continue*

Continues execution coverage session. Execution coverage can be run and results added in addition to the previous session or even to an old session being loaded into the current workspace.

### 3.5 Stop

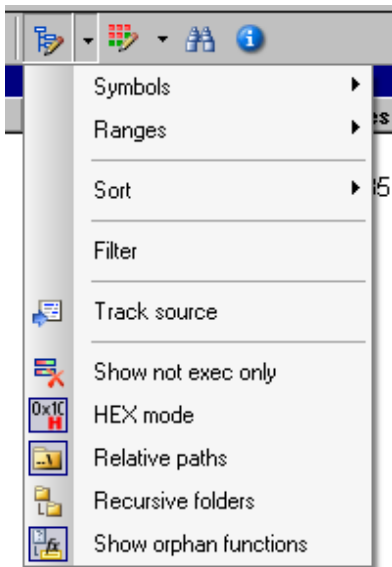
Stops execution coverage session.

### 3.6 Export



Saves results in HTML, TXT, CSV, or XML format.

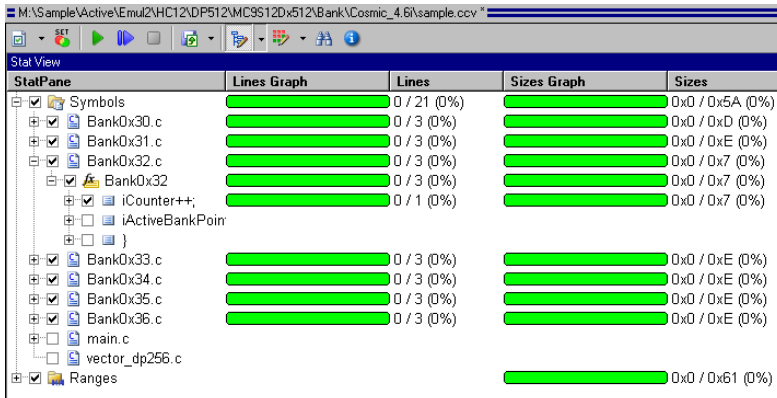
### 3.7 Statistics



The statistics toolbar covers different settings related to the displaying the coverage information.

‘Sort’ - Symbols and ranges can be sorted by different type and ordering. Clicking the column name (e.g. Lines, Sizes,...) in the execution coverage window performs sorting too.

‘Filter’ –When selected, square boxes become displayed next to the results displayed in the tree structure. By clicking in the box a tick is displayed or removed for individual coverage result. Filter acts on collected execution coverage results and has nothing to do with the initial coverage configuration, where regions/objects to be tested are defined.



StatPane	Lines Graph	Lines	Sizes Graph	Sizes
<input checked="" type="checkbox"/> Symbols		0 / 21 (0%)		0x0 / 0x5A (0%)
<input checked="" type="checkbox"/> Bank0x30.c		0 / 3 (0%)		0x0 / 0xD (0%)
<input checked="" type="checkbox"/> Bank0x31.c		0 / 3 (0%)		0x0 / 0xE (0%)
<input checked="" type="checkbox"/> Bank0x32.c		0 / 3 (0%)		0x0 / 0x7 (0%)
<input checked="" type="checkbox"/> Bank0x32		0 / 3 (0%)		0x0 / 0x7 (0%)
<input checked="" type="checkbox"/> iCounter++;		0 / 1 (0%)		0x0 / 0x7 (0%)
<input type="checkbox"/> iActiveBankPoin				
<input type="checkbox"/> }				
<input checked="" type="checkbox"/> Bank0x33.c		0 / 3 (0%)		0x0 / 0xE (0%)
<input checked="" type="checkbox"/> Bank0x34.c		0 / 3 (0%)		0x0 / 0xE (0%)
<input checked="" type="checkbox"/> Bank0x35.c		0 / 3 (0%)		0x0 / 0xE (0%)
<input checked="" type="checkbox"/> Bank0x36.c		0 / 3 (0%)		0x0 / 0xE (0%)
<input type="checkbox"/> main.c				
<input type="checkbox"/> vector_dp256.c				
<input checked="" type="checkbox"/> Ranges				0x0 / 0x61 (0%)

'Track source' - It jumps to the source of the selected symbol. The alternative is double click on the symbol.

'Show not executed' - When selected, it displays not executed parts of the code only.

'HEX mode' - Displays numbers in a hexadecimal format instead of a default decimal.

'Relative paths' - Displays relative paths next to the symbol names instead of absolute paths.

### **3.8 *Memory map***

Statement coverage results can be seen also from the memory map point of view.

### **3.9 *Search***

Search can be used to find a symbol in the results.

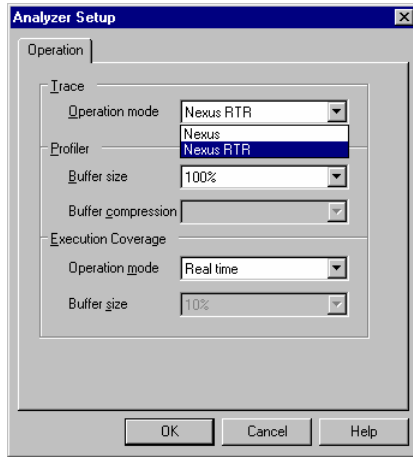
### **3.10 *Information***

The user can enter tester name, test ID, time, date, environment information and comments as part of the execution coverage session.

## 4 Hardware Configuration

Real-time or off-line coverage is selected in the 'Hardware/Analyzer Setup' dialog. Real-time execution coverage is not available for all CPU architectures featuring on-chip debug trace solution.

- Freescale MPC5500

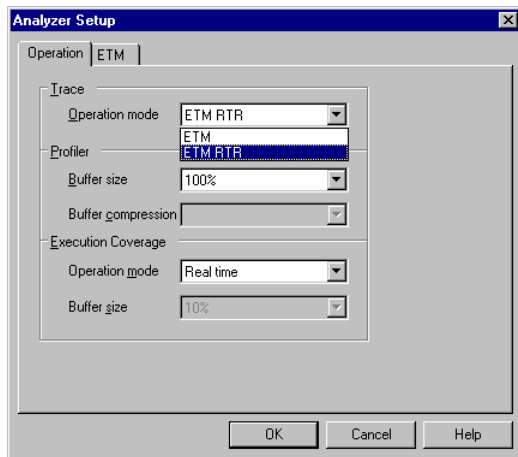


Real-time execution coverage is configured by the trace operation mode set to 'Nexus RTR'. This setting sets default 'Real time' execution coverage operation mode.

Off-line execution coverage is configured by the trace operation mode set to 'Nexus', which defaults to off-line execution coverage operation mode.

Note: Nexus RTR trace operation mode is not available for MPC551x CPUs (e.g. MPC5516) and consequentially real-time execution coverage is not available.

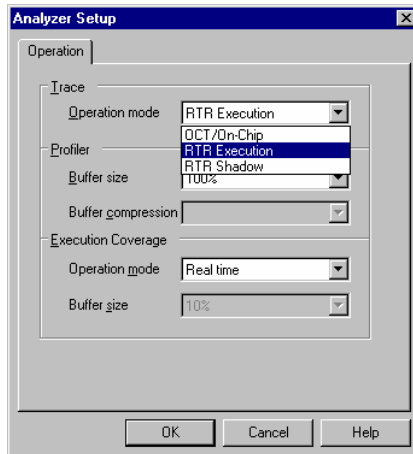
- ARM ETM



Real-time execution coverage is configured by the trace operation mode set to 'ETM RTR'. This setting sets default 'Real time' execution coverage operation mode.

Off-line execution coverage is configured by the trace operation mode set to 'ETM', which defaults to off-line execution coverage operation mode.

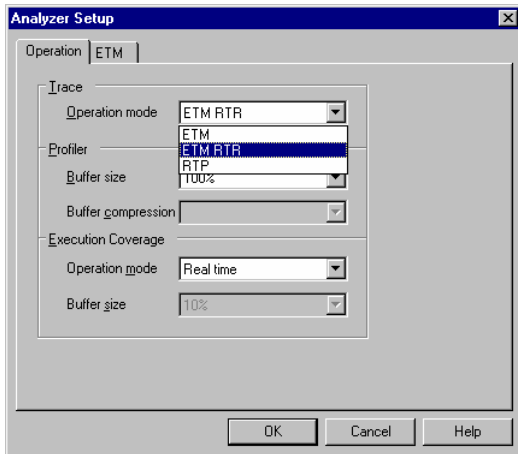
- NEC V850ES/Fx3 ActivePRO POD



Real-time execution coverage is configured by the trace operation mode set to 'RTR Execution'. This setting sets default 'Real time' execution coverage operation mode.

Off-line execution coverage is configured by the trace operation mode set to 'OCT/On-Chip', which defaults to off-line execution coverage operation mode.

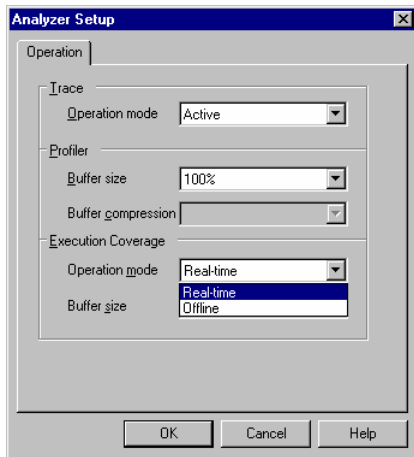
- TMS470Pxx Standard ActivePRO POD



Real-time execution coverage is configured by the trace operation mode set to 'ETM RTR'. This setting sets default 'Real time' execution coverage operation mode.

Off-line execution coverage is configured by the trace operation mode set to 'ETM', which defaults to off-line execution coverage operation mode.

- Active PRO/GT PODs (in-circuit emulators with bus trace)



Trace operation mode is set to 'Active'.

Execution coverage type is configured by selecting 'Real-time' or 'Off-line' execution coverage operation mode. Off-line operation is supported on HCS12 Active and S12X ActivePRO PODs only..

## 5 Getting Started

1. Configure real-time or off-line execution coverage in the 'Hardware/Analyzer Setup'.
2. Open execution coverage window either from 'View/Execution Coverage' or from 'File/New/Execution Coverage File (.ccv)'
3. Configure areas to be tested for the statement and/or decision coverage. Program areas covered by the execution coverage can be configured through 'Coverage configuration' use or 'Configure rangeset' use or as a mixture of both. When areas are configured, debug download must be performed first.
4. Execute debug reset, start execution coverage and run the application. Stop the application or execution coverage to view the results.

---

Disclaimer: iSYSTEM assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information herein.

© iSYSTEM. All rights reserved.