Performance Analysis

The real-time performance analyzer can display the percentage of time spent by a program in different functions or modules of a program. This functionality isn't available on the HAC. On the SA120 this analyzer can be connected to the levels or flags of the trigger unit too, to show their time relations.

Analyzer Structure

The standard measurement method for TRACE32-ICE is **Hardware**.

If the measurement method Hardware is use, the performance analyzer can be seen as an array of time counters. Only one counter is getting the clock to count, all other counters are stopped. The resolution of the counters is 1 us. Counter clocks can be enabled either by the six upper breakpoint bits (Spot, Read, Write, Alpha, Beta, Charly), the level of the trigger unit (STU) or the flags of the trigger unit (only SA120).

When using the breakpoint bits, either all CPU bus cycles can change the selection of the counter, or the program fetch cycles changes the counter selection only. The first case is used for memory access profiling, the second one for performance analysis.

```
<table>
<thead>
<tr>
<th>selection signal (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>breakpoints or</td>
</tr>
<tr>
<td>levels or flags</td>
</tr>
<tr>
<td>counter</td>
</tr>
<tr>
<td>counter</td>
</tr>
<tr>
<td>pass counter</td>
</tr>
<tr>
<td>DEMUX</td>
</tr>
<tr>
<td>time-clock</td>
</tr>
<tr>
<td>1 us</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>time counter</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>(64/32 counters)</td>
</tr>
</tbody>
</table>
```

Twin counters are used, to read and display the results without losing time information. The entry of a counter level is sampled by an extra ‘pass’ counter (only SA120).

The measurement is made in intervals, after each interval the results in the display windows are refreshed. The default interval is 1 s. All results can be displayed for the complete measurement, or for the last interval (dynamic display). As the time clock stops when the program stops, it is also possible to measure only one short run of the target program, maybe one pass of the main loop.
The address ranges can be defined either by entering the function names or address ranges in a definition window, or by an automatic programming system, which can be based on the symbol tables of the program or in a fixed address raster (screening). The address range of this functions can be limited by `PERF.Address`. If the number of symbols exceeds the number of counters (64 on SA120/HA120, 32 on ECC8), the analyzer will go into a scanning mode. There are two different scanning modes: address scanning and performance scanning. The modes are distinguished by the selection of the `PERF.Sort` command. If the sorting is OFF, the address scanning is activated, this allows scrolling through the entire symbol list. The analyzer programs only a part of the symbols and proceeds to the next symbols when required. If Ratio sort is selected, the performance scanning will be active. In this mode the analyzer programs the upper half of its counter array to a part of the list, makes a measurement, sorts the results by 'ratio' and continues with the next symbols. As a result the analyzer filters the most time consuming functions in the lower half of the counters (the first 16 or 32).

The trigger system and spot system are turned off, when the performance analyzer is active. The breakpoints are reserved for the performance analyzer during the measurement.

### Commands

The following commands define the operation mode of the performance analyzer.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PERF.state</code></td>
<td>Display the control window</td>
</tr>
<tr>
<td><code>PERF.Mode</code></td>
<td>Selects the operation mode of the performance analyzer</td>
</tr>
<tr>
<td><code>PERF.AnyAccess</code></td>
<td>Selects memory access profiling mode instead of performance analyzer mode</td>
</tr>
<tr>
<td><code>PERF.PreFetch</code></td>
<td>Make covered ranges smaller, to ignore prefetched cycles</td>
</tr>
<tr>
<td><code>PERF.Entry</code></td>
<td>Split ranges to measure the average function execution time</td>
</tr>
<tr>
<td><code>PERF.Init</code></td>
<td>Clear the time counters of the performance analyzer</td>
</tr>
<tr>
<td><code>PERF.Address</code></td>
<td>Limits the address range for automatic programmed modes</td>
</tr>
<tr>
<td><code>PERF.Gate</code></td>
<td>Define measurement intervals</td>
</tr>
<tr>
<td><code>PERF.Sort</code></td>
<td>Define display sort order and scanning modes</td>
</tr>
<tr>
<td><code>PERF.RESet</code></td>
<td>Restore all settings to the default</td>
</tr>
</tbody>
</table>
All functions of the performance analyzer can be controlled by the performance analyzer state window:

For most applications it is sufficient to select one of the predefined modes and display the results by a performance list window. The Function mode will analyze the time spent in different functions. The covered address range can be limited to one module by the PERF.Address command. The PERF.Sort will determine the display of the results.
Display Results

The results can be displayed in different formats.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERF.List</td>
<td>Display the results in a table format</td>
</tr>
<tr>
<td>PERF.View</td>
<td>Display all the result values of one address range</td>
</tr>
<tr>
<td>PERF.Sort</td>
<td>Define display sort order</td>
</tr>
</tbody>
</table>

### Release List

```
E::PERF.List
1 2 5 10 20 50 100
symbolname  ratio
            -------
(other)     43.001%  
func0       0.000%   
func1       1.056%   
func2       4.357%   
func3       0.044%   
func4       0.968%   
func5       0.396%   
func6       1.584%   
func7       2.068%   
func8       9.991%   
func9       2.772%   
func10      20.158%  
```

Manual Address Selection

Besides the automatic modes, it is possible to define the covered address ranges manually. The ranges are entered in a text file. The file can consist of function names, module names or regular address ranges.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERF.Program</td>
<td>Interactive definition of the monitored address ranges</td>
</tr>
<tr>
<td>PERF.ReProgram</td>
<td>Load the definition of the monitored address ranges from a batch scripts</td>
</tr>
</tbody>
</table>