(54) Title: DYNAMIC TRACING IN A REAL-TIME SYSTEM

(57) Abstract: The present invention relates to a method for tracing in a running real-time system. The system comprises programming code by which traffic dialogues are executed. Trace points are defined in the code. The method comprises the following steps: specifying in a trace bullet object, a trace procedure. Activating in an argument list, trace for at least one selected dialogue. The selected dialogue encounters a trace point. Information is requested whether the trace bullet object is activated for the selected dialogue or not. Confirmation that the trace is activated is sent from the argument list to the trace point.
Declaration under Rule 4.17:
— of inventorship (Rule 4.17(iv)) for US only

Published:
— with international search report
DYNAMIC TRACING IN A REAL-TIME SYSTEM

TECHNICAL FIELD OF THE INVENTION

The present invention relates to a method for dynamic tracing in a running real-time system, without affecting the systems normal performance, characteristics and behaviour.

DESCRIPTION OF RELATED ART

Most real-time systems are difficult to perform tracing, profiling and diagnostics within. This is a natural property of real-time systems since they are designed to be aware of time and deal with e.g. time-delays. Therefore it is commonly quite difficult to perform different kinds of tracing, profiling and diagnostics without affecting the systems normal performance, characteristics and behaviour.

Using different kinds of built-in trace printout functions most commonly performs tracing abilities in real-time systems. However, a trace printout function is normally activated within the scope of the whole system. So, the scope of the built-in function is static and will produce the same amount of data each time it is used, and always for all parts of the system that supports the trace function.

Using commercial tools designed specially for this purpose normally does profiling in real time systems. However, they are often limited to only be able to profile a portion of the system each time, and not the complete system.

Diagnostics (and also debugging) in real-time systems is normally done by using commercial tools designated specially for this purpose. However, they are often limited to only be able to produce reliable data during a short time span. The reason for this is that the task very often introduce delays and changes that will affect the systems normal performance, characteristics and behaviour.
Often tracing, profiling and diagnostics tasks are activated for a single-dialogue load. This since the person performing the task is only focusing on one single issue and only wants data about e.g. a specific dialogue. The task result will thereby only be based on this load profile. However, errors in a real-time system often are so-called timing problem, which only occurs during heavy load by many dialogues. If activating e.g. tracing in a system handling a greater dialogue load, the trace will not be accurate. The tracing function itself may then change the system performance, characteristics and behaviour. Further, the interpretation of all trace printouts is quite difficult to handle if using a high load.

In the US patent US 5,737,521 a trace system for analysing errors in running real-time systems is disclosed. In the US patent an activation device effects execution of predetermined functions at predetermined program points, so called trace points. In the US patent, trace procedures are defined in close relation to the code in which tracing is performed which makes the trace function static due to difficulties in modifying the code. The built-in function according to the US patent will produce data for all traffic dialogues passing those parts of the system that supports the trace function. This leads to a high distribution of trace load, load that affects the systems normal performance.

**SUMMARY OF THE INVENTION**

The present invention relates to a problem how to avoid unnecessary unwanted information after activation of a trace. Such unnecessary information is time consuming and affects the systems normal performance, characteristics and behaviour. The information also often has to be filtered out in a later stage.
A further problem solved by the invention is avoidance of trace profiles defined in close relation to the code in which tracing is performed. This makes it difficult to change and update a trace procedure and will lead to a more static behaviour.

A purpose with the invention is to invoke tracing, profiling and diagnostics on a single dialogue during normal or heavy load of the system, and enable dynamic trace, profiling and analysis of a single dialogue with a minimum of influence on systems performance, characteristics and behaviour.

The problem is solved by the invention by attaching a trace profile to at least one selected dialogue executed in the real-time system.

More in detail, the invention presents a method for tracing in a running real-time system, by specifying in a trace bullet object, a trace procedure, and by activating in an argument list a trace specification attached to at least one selected dialogue executed in the real-time system. When the selected dialogue passes a trace point, an inquiry is performed requesting whether the trace specification is activated for the selected dialogue or not. If not activated, the dialogue will just pass the trace point. If activated, after confirmation that the trace was activated for the selected dialogue, the trace procedure will be performed.

An advantage with the invention is that it provides easier fault analysis, more easily gained overall knowledge of e.g. a function reported faulty, and also a very good base for profiling of the system.

Another advantage with the invention is that it will provide to a very little cost, a very easy tailor-made trace, profiling and diagnostics functionality within the normal run-time-environment.
Yet another advantage is that the suggested trace pattern may also be used to realise other functionality if desired.

The invention will now be described more in detail with the aid of preferred embodiments in connection with the enclosed drawings.

**BRIEF DESCRIPTION OF THE DRAWINGS**

Figure 1 is a block schematic illustration of a real-time system in which two dialogues are executed.

Figure 2 shows programming code divided into modules, wherein the two dialogues are executed.

Figure 3 is a flow chart illustrating some of the most essential steps of the inventive method.
DETAILED DESCRIPTION OF EMBODIMENTS

Figure 1 discloses a running real time system RTS. The system according to this example comprises a message gateway module MGM, dialogue handling module systems DHMS1 and DHMS2, and message handling module systems MUMS1:1, MUMS1:2, MUMS2:1 and MUMS2:2. External dialogue parts EDP1 and EDP2 can be seen in figure 1 and are arranged to communicate with the real time system RTS. A database DB is used to store traffic handling data. The database is in this example located outside the real time system RTS but might as well be located inside. A dialogue is defined as a communication sequence used to perform a specified task, from initiation of the task until it is finished. An example of a task might be the setting up of a mobile call. Two dialogues, a first dialogue D1 and a second dialogue D2 are referred to in the figure. The first dialogue D1 is started up when a message arrives to the management gateway module MGM from the external dialogue part EDP1. As can be seen in the figure, the first dialogue D1 performs bi-directional communications in the real-time system between the entities MGM, DHMS1, MUMS1:1, MUMS1:2 and DB. This is continued until the specified task is completed. During the whole process, the first dialogue follows a route, which is shown in figure 1 by the references D1. In the same way, the second dialogue D2 follows a route shown in figure 1 by the references D2. The different entities are built up by programming code divided into modules. Each module might represent an entity, but this can vary depending on the type of code that is used. During execution of the dialogues D1 and D2, the routes pass through different modules. In the following example, which now will be shown in figure 2, trace points have been defined in some of those modules. During the execution, the dialogues D1 and D2 will pass the trace points. It is to be noted that this is just an example and
often trace points are located in all modules and not only in some of the modules in the code.

Figure 2 discloses in a first embodiment of the invention, the mentioned programming code CODE used to build up the real time system. The code has already been shown in figure 1. The code is divided into different modules M1-M6. Trace points TP1-TP5 are defined in some of the modules so that a dialogue will enter a trace point as soon as it enters into a module wherein a trace point is defined. An argument list CO1 and a trace bullet object CO2 are arranged to communicate with the trace points TP1-TP5. Both the argument list CO1 and the trace bullet object CO2 are each one shown in figure 1 with more than one reference symbol, although only one of each entity actually exists in the system in this exemplifying embodiment. This is just for the sake of clarity of the figure. Both the arrangement list CO1 and the trace bullet object CO2 are easily accessible by a terminal TE. A trace procedure TP, i.e. the procedure wherein a wanted trace, diagnostics or profiling is specified, is stored in the trace bullet object CO2 with the aid of the terminal TE. A desired trace might for example be just the presence of a dialogue in a module. A trace might also be more advanced, tracing more specifically in detail what is happening after a dialogue has entered the module. The argument list CO1 comprises a specification for which dialogue, or for which dialogues, tracing is desired.

The invention will now be explained more in detail by the aid of a method according to the invention. The two dialogues D1 and D2, shown in figure 1, are in the method below, after execution, followed step by step through parts of the code shown in figure 2. The two dialogues might be executed simultaneously but in the described method below, the first dialogue D1 is first executed and followed step by step. Thereafter, the second dialogue will be executed and followed step by step. In the example, before the actual
execution of the dialogues, it is specified in the argument list CO1, that trace is desired for the first dialogue D1 only. A specification of the desired trace is stored in the trace bullet object CO2. The trace specification in this example is just a report of the presence of a dialogue in a module, i.e. the presence of the dialogue specified in the argument list. The specified dialogue in this example was the first dialogue D1. In addition to the steps already mentioned, the following steps will disclose the invention:

- The first dialogue D1 is started. The actual start up of the dialogue happens after a message requesting the dialogue, arrives to the message gateway module MGM from the external dialogue part EDP1. The message gateway module MGM and the external dialogue part EDP1 have both been shown in figure 1. In figure 2, the start of the first dialogue can be seen in module M1.

- The first dialogue D1 enters an exit of module M1. Exactly what has happened in module M1 before the exit is of less importance to the explanation of the invention and will therefore not be further analysed.

- The first dialogue makes a program jump from module M1 to module M2. The jump has been triggered by a program instruction in module M1. The program jump is illustrated in figure 2 by the reference D1:J1.

- The first dialogue D1 enters into module M2 and a trace point TPT1 is executed.

- Information is required by module M2, whether the trace is specified as desired for the first dialogue D1. Sending a signal 11 from the module M2 to the argument list CO1 requires this information. The code implementation is of course dependent on what programming language that is used. The step "sending a signal" may in real be realised
by for example an IF-sentence checking if trace is activated for the dialogue of interest or not.

- Sending a signal 12 from the argument list CO1 to the module M2 confirms that trace was activated for the first dialogue D1.

- A flag ENTRY is set in the trace bullet object CO2 when the programming instructions in the module M2 are to be handled by the trace bullet object CO2.

- A call back procedure 13 between the trace bullet object CO2 and different method parts in the module M2 via the trace point TPT1 is performed. The call back procedure is the trace procedure TP specified in the trace bullet object CO2. In this example it just shows the presence of the dialogue in module M2. As an alternative it might have been specified to involve steps like fetching different variables generated by the first dialogue and print these variables.

- A flag EXIT is set in the trace bullet object CO2 when the programming instructions in the module M2 no longer are handled by the trace bullet object CO2. The implementation to use the different flags ENTRY and EXIT is of course dependent on the type of programming language that is used. Another possible implementation would be to call entry or exit methods within the trace bullet object whenever the trace bullet object is entered or abandoned.

- The first dialogue D1 makes a program jump from module M2 to module M5. The flag EXIT is set in the trace bullet object CO2 before the jump takes place. The program jump which is part of the dialogues normal executive phase is illustrated in figure 2 by the reference D1:J2. In module M5, the dialogue D1 encounters a trace point TPT4. The same procedure as described above is then performed in module M5. This has been disclosed in figure 2 by the
signals 14 and 15 and by the call back procedure 16 between the module M5 and the trace bullet object CO2. The dialogue thereafter continues its route through the code until it has finished its task whereby the external dialogue part EDP1 is informed that the task is completed.

- The second dialogue D2 is started. The actual start up of the dialogue happens after a message requesting the dialogue, arrives to the message gateway module MGM from the external dialogue part EDP2.

- The second dialogue enters the exit of module M1.

- The second dialogue makes a program jump from module M1 to module M2. The program jump is illustrated in figure 2 by the reference D2:J1.

- The second dialogue D2 enters into module M2 and a trace point TPT1 is executed.

- Information is required by module M2, whether the trace is specified as desired for the second dialogue D2. Sending a signal 21 from the module M2 to the argument list CO1 requires the information.

- Sending a signal 22 from the argument list CO1 to the module M2 says that trace was not activated for the second dialogue D2.

- The second dialogue D2 makes a program jump from module M2 to module M6. The program jump is illustrated in figure 2 by the reference D2:J2. The dialogue then continues its route through the code until it has finished its task whereby the external dialogue part EDP2 is informed that the task is completed. It is to be noted that the dialogue D2 passes all trace points without taking any further measures since trace was not activated for the second dialogue.
The inventive method to perform tracing, profiling or diagnostics in a real-time system has been discussed above. In figure 3, some of the most essential steps of the method are shown in a flow chart. The already described figures 1 and 2 are to be read together with figure 3. The method disclosed in figure 3 comprises the following steps:

- It is specified in the argument list CO1 that trace is desired for the first dialogue D1. A specification of the desired trace is stored in the trace bullet object CO2. This is disclosed in figure 3 with a block 100.

- The first dialogue D1 is started. This is disclosed in figure 3 with a block 101.

- The first dialogue D1 makes a program jump D1:J1 from module M1 to module M2. This is disclosed in figure 3 with a block 102.

- The first dialogue D1 enters into module M2 and the trace point TPT1 is executed. This is disclosed in figure 3 with a block 103.

- Sending of the signal 11 from the module M2 to the argument list CO1 requires information whether the trace is specified as desired for the first dialogue D1. This is disclosed in figure 3 with a block 104.

- Sending the signal 12 from the argument list CO1 to the module M2 confirms that trace was activated for the first dialogue D1. This is disclosed in figure 3 with a block 105.

- The call back procedure 13 between the trace bullet object CO2 and method parts in M2, via the trace point TPT1 is performed. This is disclosed in figure 3 with a block 106.

Depending on the desired levels of trace/profile/diagnostics etc, different implementations of trace procedures TP may be
sent into the system. The design of the trace procedure determines what methods to call and the amount of data to be printed, and also what data to print. One trace procedure may be designed for profiling purposes while another is tailor-made for finding some odd behaviour in the system whereby another more specific procedure is defined, concentrated to the parts showing the odd behaviour. The only limitation is set by the methods that are available to use in each object. If combining this pattern with generation of additional methods into the source code, one could realise a lot of other functions. One could, e.g. use this pattern to implement functions as TelORB-, FORLOPP- or RED LINE-trace. The design pattern is not restricted or limited to a certain usage. In other words, the invention is not limited to the above described and in the drawings shown embodiments but can be modified within the scope of the enclosed claims.
CLAIMS

1. Method for tracing in a running real-time system comprising programming code (CODE) by which dialogues (D1, D2) are executed, in which code trace points (TPT1-TPT5) are defined, which method comprises the following steps:

   - specifying in a trace bullet object (CO2), a trace procedure (TP), whereby the method is characterised by the following steps:

   - activating in an argument list (CO1), trace for at least one selected dialogue (D1);

   - arrival (D1:J1) by the selected dialogue (D1), to a trace point (TPT1);

   - sending from the trace point (TPT1) into the argument list (CO1), an inquiry (11) requesting information whether trace is activated for the selected dialogue (D1).

2. Method for tracing in a running real-time system according to claim 1, which method comprises the following further step:

   - sending from the argument list (CO1) to the trace point (TPT1), a confirmation (12) that trace is activated.

3. Method for tracing in a running real-time system according to claim 2, which method comprises the following further step:

   - performance of the specified trace procedure (TP) by executing a call back procedure (13) between programming instructions in the code (CODE) referred to by the trace point (TPT1), and the trace bullet object (CO2).
4. Method for tracing in a running real-time system according to claim 3, which method comprises the following further steps:

- informing the trace bullet object (CO2) that programming instructions in the code (CODE) referred to by the trace point (TPT1) at present is handled by the trace bullet object (CO2).

- informing the trace bullet object (CO2) that programming instructions in the code (CODE) referred to by the trace point (TPT1) at present not is handled by the trace bullet object (CO2).

5. Method for tracing in a running real-time system according to claim 4, which method comprises the following further steps:

- setting a flag ENTRY in the trace bullet object (CO2) when the programming instructions in the code referred to by the trace point (TPT1) are to be handled by the trace bullet object (CO2);

- setting a flag EXIT in the trace bullet object (CO2) when the programming instructions in the code referred to by the trace point (TPT1) no longer are handled by the trace bullet object (CO2).

6. Method for tracing in a running real-time system according to any of claim 1-5, whereby different kinds of procedures can be specified in the trace procedure (TP) such as:

- tracing;

- diagnostics;

- profiling.
7. Method for tracing in a running real-time system according to any of claim 1-6, whereby the trace bullet object (CO2) is located remotely from the code (CODE).

8. Method for tracing in a running real-time system according to any of claim 1-7, whereby the argument list (CO1) is located remotely from the code (CODE).

9. Method for tracing in a running real-time system according to claim 1, which method comprises the following step:

- sending from the argument list (CO1) to the trace point (TPT1), a message that trace is not activated for the selected dialogue (D1).

10. Method for tracing in a running real-time system comprising programming code (CODE) according to any of claim 1-9, which code is divided into programming modules (M1-M6) whereby trace points (TPT1-TPT5) are defined as to be entered by a dialogue (D1, D2) when the dialogue enters into a new module (M1-M6).
Fig. 1
Fig. 2
A specification that trace is desired for D1 is stored in CO1. A definition of the trace is stored in CO2.

The first dialogue D1 is started.

D1 makes a jump D1:J1 into module M2.

Trace point TPT1 is executed.

Information is required whether trace is activated for the first dialogue D1.

Trace activation for D1 is confirmed.

The trace procedure TP is performed.

Fig. 3
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 11/00
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE, DK, FI, NO classes as above

Electronic database consulted during the international search (name of database and, where practicable, search terms used)

EPO-INTERNAL, WPI DATA, PAJ, INSPEC, COMPENDEX, TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>EP 0801348 A1 (HEWLETT-PACKARD COMPANY), 15 October 1997 (15.10.97), page 2, line 42 - page 3, line 3; page 3, line 57 - page 4, line 1; page 4, line 12 - line 16, page 10, line 6; page 7, line 9 - line 10; page 3, line 39 - line 46, claims 6, 7, 8</td>
<td>1-10</td>
</tr>
<tr>
<td>A</td>
<td>US 5946486 A (PEKOWSKI, R.P.), 31 August 1999 (31.08.99), column 2, line 10 - column 3, line 59</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>A</td>
<td>WO 9845784 A1 (ADVANCED MICRO DEVICES, INC.), 15 October 1998 (15.10.98), claims 1-5</td>
<td>3, 4, 5, 7</td>
</tr>
</tbody>
</table>

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

- “A” - document defining the general state of the art which is not considered to be of particular relevance
- “E” - earlier application or patent but published on or after the international filing date
- “L” - document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- “G” - document referring to an oral disclosure, use, exhibition or other means
- “P” - document published prior to the international filing date but later than the priority date claimed

“T” - later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“Y” - document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” - document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” - document member of the same patent family

Date of the actual completion of the international search: 12 February 2002

Date of mailing of the international search report: 05-04-2002

Form PCT/ISA/210 (second sheet) (July 1998)
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US 5737521 A (KLEBER, U. ET AL), 7 April 1998 (07.04.98), whole document</td>
<td>1-10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP 0801348 A1</td>
<td>15/10/97</td>
<td>US 6163881 A</td>
<td>19/12/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 5946486 A</td>
<td>31/08/99</td>
</tr>
<tr>
<td>WO 9845784 A1</td>
<td>15/10/98</td>
<td>DE 69801156 D</td>
<td>00/00/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 69801220 D</td>
<td>00/00/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 0974093 A,B</td>
<td>26/01/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 0974094 A</td>
<td>26/01/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 0974096 A,B</td>
<td>26/01/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2001519947 T</td>
<td>23/10/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2001519948 T</td>
<td>23/10/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2001519949 T</td>
<td>23/10/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 5978902 A</td>
<td>02/11/99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6009270 A</td>
<td>28/12/99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6041406 A</td>
<td>21/03/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6094729 A</td>
<td>25/07/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6142683 A</td>
<td>07/11/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6148381 A</td>
<td>14/11/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6154856 A</td>
<td>28/11/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6154857 A</td>
<td>28/11/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6167536 A</td>
<td>26/12/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6185732 B</td>
<td>06/02/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6189140 B</td>
<td>13/02/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6314530 B</td>
<td>06/11/01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 9845782 A</td>
<td>15/10/98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 9845783 A</td>
<td>15/10/98</td>
</tr>
<tr>
<td>US 5737521 A</td>
<td>07/04/98</td>
<td>AT 153786 T</td>
<td>15/06/97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2172737 A</td>
<td>06/04/95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE 59402939 D</td>
<td>00/00/00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DK 721620 T</td>
<td>08/12/97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 0721620 A,B</td>
<td>17/07/96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SE 0721620 T3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ES 2102881 T</td>
<td>01/08/97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GR 3024304 T</td>
<td>31/10/97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 9503084 T</td>
<td>25/03/97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 9509396 A</td>
<td>06/04/95</td>
</tr>
</tbody>
</table>