MCR10096
Fix fault_tag_1 fault in TOLTS
Revision 1.0

Eric Swenson

October 24, 2021

1 Problem Description

When running TOLTS (bound_tolts_$tolts_), TOLTS test pages can generate lockup faults. These occur due to `tze *,ic` instructions in the test pages for situations not anticipated by the test code. It is very likely that the unexpected failure paths were not exercised during the original lifetime of Multics, and therefore not handled. The use of the `tze *,ic` instruction was likely done so that Honeywell field engineers (FEs), with access to the microfiche for the TOLTS test pages, could examine the IC at the time of the lockup fault, find the corresponding offset in the microfiche, and determine the failure.

When running under the DPS8/M simulator, where not all I/O commands are yet supported, it is possible for I/O commands to return results not anticipated or handled by TOLTS. In these cases, TOLTS test pages may generate these lockup faults. Unfortunately, due to a bug in the driver TOLTS driver, `mtdsim_`, a lockup fault received during execution results in a `fault_tag_1` fault, because the code attempts to reference through an unitialized pointer.

The output generated with the problem unfixed looks similar to this:

```
polt encountered a lockup fault a dump will be taken

Error: fault_tag_1 by mtdsim_$tolts_abort|34707
(>system_library_tools>bound_tolts_)
referencing stack_4|40040 (in process dir)
Ascii data where pointer expected.
r 11:11 9.369 344 level 2

polts dump file >user_dir_dir>SysAdmin>Swenson>!BBBKQPZpgbMlLN.polt.dump has been queued
    for printing
```

The POLTS dump file correctly identifies the `tze *,ic` instruction and the IC (in the test page) where execution stopped.

The Multics ticket for this problem is: https://multics-trac.swenson.org/ticket/251.

2 Problem Cause

On many error conditions during TOLTS execution, the local procedure `tolts_abort` is invoked. This procedure assumes that code that executed before calling this function has set the SCU data pointer (`scup`) to point to properly saved SCU data. Not all callers do this, though, and in particular, a lockup fault during the running of the test page doesn't initialize `scup`. A `lockup` condition handler is established by the `mtdsim_` driver, as are lots of other condition handlers. None of these sets the `scup` pointer. This pointer, is set, however, when the test page gets a fault NOT handled by condition handlers in `mtdsim_`, but where the fault occurs during execution of the GCOS test page code itself.

3 The Fix

The fix for this issue is quite easy. At an appropriate place in the code, the `scup` is set to `null` and before referencing through this pointer, its value is checked for `null`. There is already logic in place to handle the fact that certain errors might not have any abort address, which is what the reference through this pointer attempts to derive.

The `compare_ascii` output, below, shows the required changes.

```
Inserted in B:
B499          scup = null;
Preceding:
A497          on lockup begin;


A3222         spa.abort.add = scu.ilc;                          /* equals address of last
    mme */
Changed by B to:
B3225         if scup = null then
B3226            spa.abort.add = "0"b;                          /* don't have an address,
    got a non MME fault */
B3227         else
B3228            spa.abort.add = scu.ilc;                       /* equals address of last
    mme */

Comparison finished: 6 differences, 26 lines.
```

4 Testing

Testing of this fix requires simply running TOLTS where test pages are known to cause lockup faults. When these occur, with the fix above, TOLTS now aborts with:

```
polt encountered a lockup fault a dump will be taken

polts dump file >user_dir_dir>SysAdmin>Swenson>!BBBKQPZpgbMlLN.polt.dump has been queued
    for printing
```

5 Documentation

There are no documentation changes for this change.

6 Version History

2021-10-24	1.0	Initial version of the MCR.