MCR10107
Fix Bad Error Message in ISOLTS and Bad Error Codes Returned by configure_test_cpu
Revision 1.0

Eric Swenson

December 23, 2021

1 Problem Description

When running ISOLTS, the user must have permission to "set proc required" – that is, to specify on which processor the program should run. An access control segment (ACS), `>sc1>admin_acs>set_proc_required.acs`, is used to control access to this function. The user must have rw access to this ACS segment in order to set the required processor during testing. This ACS segment is checked by hardcore when told to set the required processor for ISOLTS testing, and if access is not granted, the supporting hardcore program, `configure_test_cpu`, returns an error code to the caller (`isolts_`), an error message is displayed, and the attempted ISOLTS run is aborted.

Due to a bug in `configure_test_cpu`, an incorrect error code is returned to `isolts_`, and ISOLTS misdiagnoses the error with the message:

```
isolts: the following errors were detected while attempting reconfiguration:

isolts: scu b has no interrupt mask register assigned to cpu b

***isolts executive version 116011 off 211223 at 13.392

***enter "polts", "molts", "colts", "isolts", "quit", or "msg"
???
```

The ticket for this issue is https://multics-trac.swenson.org/ticket/30

2 Reasons for Current Behavior

The `configure_test_cpu` program performs many checks while trying to set up the environment for CPU testing and returns a wide variety of error codes. All the error codes returned are of the form `rcerr_isolts_xxxx` – except for one – the one reporting a failure to set the required processor. There are multiple sets of error codes defined in the include file `rcerr.incl.pl1`. All the ones

handled by ISOLTS are of the form `rcerr_isolts_xxxx` and the integer value of the erorr code is used to index into an array of error messages, such as the one reported above. That message is associated with the error code `rcerr_isolts_no_mask`, whose integer value is 17:

```
dcl (rcerr_isolts_locked init (1),                    /* reconfig_lock locked to another
    process */
    rcerr_isolts_illegal_cpu init (2),                /* illegal cpu tag */
    rcerr_isolts_cpu_online init (3),                 /* requested cpu is online */
    rcerr_isolts_no_config init (4),                  /* requested cpu is not configured */
    rcerr_isolts_two_scu init (5),                    /* Must have at least two SCUs to
        run ISOLTS */
    rcerr_isolts_illegal_scu init (6),                /* illegal scu tag */
    rcerr_isolts_bootload_scu init (7),               /* requested scu is the bootload
        memory */
    rcerr_isolts_scu_not init (8),                    /* requested scu is not configured */
    rcerr_isolts_not init (9),                        /* requesting process is not ISOLTS
        process */
    rcerr_isolts_wrong_cell init (10),                /* interrupt answered in correct scu
        but wrong cell */
    rcerr_isolts_wrong_scu init (11),                 /* interrupt answered in wrong scu */
    rcerr_isolts_wrong_scu_cell init (12),            /* interrupt answered in wrong scu
        on wrong cell */
    rcerr_isolts_no_response init (13),               /* No response to a processor start
        interrupt */
    rcerr_isolts_bad_switches init (14),              /* read switch data is not in
        expected format */
    rcerr_isolts_lda_fail init (15),                  /* A LDA 2 did not operate correctly
        */
    rcerr_isolts_no_str_flt init (16),                /* No store falt when a LDA 64k was
        executed */
    rcerr_isolts_no_mask init (17)                    /* No mask set for test cpu */
    ) fixed bin static options (constant);
```

The reason why error code 17 is returned, is because the code in `configure_test_cpu.pl1` looks like:

```
/* Force our process to run on the active CPU */

         cpu_mask = "0"b;
         substr (cpu_mask, scs$processor_test_data.mask_cpu + 1, 1) = "1"b;
         call set_procs_required (cpu_mask, tcode);
         if tcode ^= 0 then do;
              rcode = rcode = rcerr_sprq_failed;
              return;
         end;
```

And later, this:

```
         if scs$processor_test_data.scu_state = "11"b then do; /* if running on lower
             memory */
              cpu_mask = "0"b;
```

```
            substr (cpu_mask, scs$processor_test_data.mask_cpu + 1, 1) = "1"b;
            call set_procs_required (cpu_mask, rcode); /* run on CPU with mask set */
            if rcode ^= 0 then do;
                rcode = rcerr_sprq_failed;
                return;
            end;
        end;
```

Note that the error code returned in both cases is `rcerr_sprq_failed`, which is also defined in `rcerr.incl.pl1` as seen below:

```
dcl (rcerr_locked init (12),                        /* database already locked */
    rcerr_online init (13),                         /* device already online */
    rcerr_no_config init (14),                      /* device not in configuration */
    rcerr_not_online init (15),                     /* device not online */
    rcerr_range init (16),                          /* request is out of range */
    rcerr_sprq_failed init (17)                     /* could not set CPU required */
    ) fixed bin static options (constant);
```

When this error code is returned to ISOLTS, it looks it up in its table, based on the `rcerr_isolts_xxx` error codes, and finds the message for error code 17 (`rcerr_isolts_no_mask`):

```
dcl  reconfig_err_message (18) char (64) static options (constant) init
     ("System dynamic reconfiguration in progress, try later", /* rcerr_isolts_locked */
     "cpu tag ^a is illegal",                       /* rcerr_isolts_illegal_cpu */
     "cpu ^a is online and unavailable for test",   /* rcerr_isolts_cpu_online */
     "cpu ^a is not configured",                    /* rcerr_isolts_no_config */
     "there must be at least two online scus to run isolts", /* rcerr_isolts_two_scu */
     "scu tag ^a is illegal",                       /* rcerr_isolts_illegal_scu */
     "scu ^a is the bootload scu and cannot be used for testing", /*
         rcerr_isolts_bootload_scu */
     "scu ^a is not online",                        /* rcerr_isolts_scu_not */
     "requesting process is not running isolts",    /* rcerr_isolts_not */
     "cpu ^a responded to interrupt cell ^a at loc ^a", /* rcerr_isolts_wrong_cell */
     "cpu ^a responded to an interrupt cell ^a on scu ^a", /* rcerr_isolts_wrong_scu */
     "cpu ^a responded to an interrupt cell ^a on scu ^a at loc ^a", /*
         rcerr_isolts_wrong_scu_cell */
     "cpu ^a failed to respond to an interrupt cell ^a interrupt", /*
         rcerr_isolts_no_response */
     "the following switches on cpu ^a are set incorrectly: ^a", /*
         rcerr_isolts_bad_switches */
     "a ""lda 2"" did not operate properly",        /* rcerr_isolts_lda_fail */
     "a ""lda 65536"" (64k) failed to produce a store fault", /* rcerr_isolts_no_str_flt */
     "scu ^a has no interrupt mask register assigned to cpu ^a"; /* rcerr_isolts_no_mask */
```

The result is the misreported error.

3 Proposed Fix

The proposed fix is first, to define a new ISOLTS error code for the `set_procs_required` error case.

```
cpa [lpn rcerr.incl.pl1] ==

A73          rcerr_isolts_no_mask init (17)                  /* No mask set for test cpu
    */
Changed by B to:
B72          rcerr_isolts_no_mask init (17),                 /* No mask set for test cpu
    */
B73          rcerr_isolts_sprq_failed init (18)              /* could not set CPU
    required */
```

And to then change `configure_test_cpu` to use that error code in the places where it used the non-ISOLTS error code before:

```
cpa [lpn configure_test_cpu.pl1] ==

A199                  rcode = rcerr_sprq_failed;
Changed by B to:
B199                  rcode = rcerr_isolts_sprq_failed;


A564                      rcode = rcerr_sprq_failed;
Changed by B to:
B564                      rcode = rcerr_isolts_sprq_failed;

Comparison finished: 2 differences, 4 lines.
r 18:20 0.718 13
```

Finally, this MCR proposes to change `isolts_.pl1` to know about the new error code in its mapping array of code-to-message:

```
cpa [lpn isolts_.pl1] ==

A965     dcl  reconfig_err_message (17) char (64) static options (constant) init
Changed by B to:
B965     dcl  reconfig_err_message (18) char (64) static options (constant) init


A982          "scu ^a has no interrupt mask register assigned to cpu ^a"); /*
    rcerr_isolts_no_mask */
Changed by B to:
B982          "scu ^a has no interrupt mask register assigned to cpu ^a", /*
    rcerr_isolts_no_mask */
B983          "unable to set CPU required for cpu ^a");       /* could not set CPU
    required */
```

```
A985          if ecode > 17 then                                /* if standard error code */
Changed by B to:
B986          if ecode > 18 then                                /* if standard error code */


Inserted in B:
B1025    etype (18):                                            /* rcerr_isolts_sprq_failed
    */
B1026            arg1 = tags (cpu_tag);
B1027            go to display_err;                             /* display message */
B1028
Preceding:
A1024    etype (14):                                            /* rcerr_isolts_bad_swiches
    */

Comparison finished: 4 differences, 11 lines.
```

As is probably obvious, the mapping array allows for placeholders – and for the new message, the proposed fix substitutes the CPU tag for the "â" in the message. That is handled by the changes in lines 1025-1027.

The change on line 986 distinguishes an error returned from `configure_test_cpu` as being one of the special ISOLTS error codes (1-18) and a normal Multics error code. The upper limit (18) must to be adjusted for the new upper limit.

4 Testing

First, prior to making the proposed changes, I ran ISOLTS to test one of the CPUs after first denying myself permission to "set procs required". I did this by deleting my access to `>sc1>admin_acs>set_proc_required.acs`. I noted that I got the incorrect error message, as described above.

Then, after creating a new hardcore with the required change a to `configure_test_cpu` and booting with that hardcore, I ran the ISOLTS test again. This time, I got an error from isolts indicating that it had received the new (18) error code, but didn't know how to interpret it (still using installed version of `bound_tolts_`):

```
asking operator to manually reconfigure cpu b
isolts: the following errors were detected while attempting reconfiguration:

isolts: Code 18. attempting reconfiguration

***isolts executive version 116011 off 211223 at 14.121

***enter "polts", "molts", "colts", "isolts", "quit", or "msg"
???
```

This indicates that the hardcore change was successful, and that `configure_test_cpu` was now returning the new error code.

Finally, I switched to a new version of `bound_tolts_` that included the above change, and this time, I got the correct (new) error messsage:

```
asking operator to manually reconfigure cpu b
isolts: the following errors were detected while attempting reconfiguration:

isolts: unable to set CPU required for cpu b
***isolts executive version 116011 off 211223 at 14.121

***enter "polts", "molts", "colts", "isolts", "quit", or "msg"
???
```

I also verified that when the user has the correct permissions on the ACS segment, that cpu testing proceeds correctly.

5 Documentation

There are no documentation changes required by this change.

6 Version History

2021-12-23	1.0	Initial version of the MCR.