MCR10146
Fix Issue With PRPH DSKn Card Parsing
Revision 1.0

Eric Swenson

February 8, 2025

1 Problem Description

The following, incorrect, config deck card:

```
prph dskb a 14 1 501. 4 -model 451. 4
```

causes Multics to crash on boot with an unhelpful error message:

```
get_main: Insufficient storage available for ioi_data
```

This is caused by incorrect parsing of the disk model and number of drives (due to the duplicate bogus values).

The ticket describing the original problem as reported is located here: https://multics-trac.swenson.org/ticket/173.

2 Proposed Fix

While it is not trivial to make the config deck parser, which is run very early in the BCE and Multics boot process, detect the incorrect card, it is possible to have the logic that attempts to determine the model of the disk, and thus the type of the disk, detect that a bogus value has been parsed.

This MCR proposes to update `get_io_segs.pl1` to handle a bogus disk model and issue an error message on the console, rather than use the erroneous value to attempt to allocate memory for the bad model and crash with the error it currently issues.

The fix is trivial and is shown in the `compare_ascii` output below:

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

Inserted in B:
B73      dcl      SYSERR_BAD_MODEL_STRING char (64) int static options (constant)
B74                                      init ("^a Invalid PRPH DSKn model: ^o.");
```

```
Preceding:
A73      dcl      SYSERR_QUEUE_STRING_1 char (50) int static options (constant)


A131                                    do dev_idx = 1 to hbound (MODEL, 1)
A132                                        while (prph_tap_card.group (i).model ^= MODEL
    (dev_idx));
A133                                    end;
Changed by B to:
B133                                    do dev_idx = 1 to hbound (MODEL, 1);
B134                                        if prph_tap_card.group (i).model = MODEL
    (dev_idx) then
B135                                            goto SET_DEVICE_TYPE;
B136                                    end;
B137                                    call syserr (CRASH, SYSERR_BAD_MODEL_STRING, ME,
    prph_tap_card.group (i).model);
B138     SET_DEVICE_TYPE:


Inserted in B:
B389        get_io_segs: Invalid PRPH DSKn model: XXX.
B390
B391        S:     $crash
B392
B393        T:     $init
B394
B395        M:     The prph card for the DSKn subsystem specifies a bad model.
B396
B397        A:     $config
B398
B399
B400        Message:
Preceding:
A384        get_io_segs: DSKQ mmm. < xxx. per drive, forcing DSKQ nnn.

Comparison finished: 3 differences, 23 lines.
```

The updated code doesn't simply use a bad `dev_idx` value to later choose the disk type, such as what would happen if the loop ran to completion without locating the model, but rather detects that a good model is not found, and issues a more appropriate error message.

The code that follows the new `SET_DEVICE_TYPE` label is:

```
SET_DEVICE_TYPE:
                                dev_idx = MODELX (dev_idx);
                                if number_of_sv (dev_idx) = 0 then
                                    physical_volumes = physical_volumes +
                                        prph_dsk_card.group (i).ndrives;
                                else physical_volumes =
                                        physical_volumes
                                        + (prph_dsk_card.group (i).ndrives * number_of_sv
                                            (dev_idx));
```

As can be seen, if the original code exits the loop without finding a matching model, the value of `dev_idx` will be one value greater than the upper bound of the `MODEL` array, and therefore also one value greater than the upper model of the MODELX array. As a result, a out-of-bounds memory access occurs (which is not detected) and garbage values are used for the above calculations.

With the fix in place, the system crashes with a much more helpful error message:

```
get_io_segs: Invalid PRPH DSKn model: 4.
bce (early) 1731.9: M->
```

This message points the operator in the correct direction to locate and fix the PRPH DSKn config card.

3 Documentation

Documentation for these error messages is in the form of "Message Documentation" which is generated from specially processed text in the source code that generates the message. This change, above, includes this documentation.

4 Test

The changes will be tested by running the updated code on several Multics systems with good and bad PRPH DSKn card models.

5 Version History

2025-02-08	1.0	Initial version of the MCR.