

MCR10190

Fix ALM Fault on a Decimal Field With No Digits

Revision 1.0

Eric Swenson

October 13, 2026

1 Problem Description

[Ticket #80](#) describes an issue where ALM takes an `illegal_procedure` fault, instead of reporting an error, when the operand of a `dec` pseudo-operation is not a number:

```
      dec      foo
foo:    end

alm crash1
ALM 8.14
Error: illegal_procedure condition by decevl_$|572
(>system_library_standard>bound_alm_)
referencing any_to_any_|1224 (>system_library_1>bound_library_wired_|46576)
```

A second report shows the same fault at a different location for `main`: `dec **1`:

```
Error: illegal_procedure condition by decevl_$|427
referencing any_to_any_|1206 (>system_library_1>bound_library_wired_|46560)
```

1.1 Cause

`decevl_` evaluates a decimal field for the `dec` and `dec_unal` pseudo-operations and for decimal literals. It collects the digits of the field into a decimal string (`number`) and counts them in `prec`, then calls `assign_` to convert that string, as a decimal number of precision `prec`, to a fixed or floating binary value. It stops at the first character it cannot use. If the field contains no digits at all, `prec` is 0, and `assign_` faults on a decimal number of precision 0. Gary Dixon traced the first case to the float conversion (`decevl_` line 256, with `prec = 0`).

- `dec foo`. A letter other than `a`, `b` and `L` is taken to start an exponent: `f` sets the E error flag and marks the field floating, and so do the two `os`. The field ends at the end of the line with no digits, and the float conversion faults (`decevl_|572`).
- `dec **1`. The leading blank is skipped; the `*` cannot be part of a number, so the field ends

there, with no digits, and the integer conversion faults (`decevl_|427`).

Any field without digits does the same: an empty operand (`dec` alone), a sign or a point alone (`dec -`, `dec .`), and the ASCII form without a count (`dec a`), which also passes `prec = 0` to `assign_` to get the count. Decimal literals reach `decevl_` only when they start with a digit, sign or point, so `=+`, `=.` and `=+aABC` fault too.

Ticket comment 3 suggests that a proper fix might be to rewrite `decevl_` (or larger parts of ALM) to tokenize its input. That is not needed to stop the fault or to diagnose the error: a field with no digits can never be a valid number, and `decevl_` already has a path that reports a bad field and returns a value so that assembly continues.

2 Proposed Fix

Before calling `assign_`, `decevl_` checks `prec`. If it is 0, it goes to a new label, `no_digits`, at the existing size-condition exit, which sets the E flag (“malformed expression in arithmetic field”, fatal), and returns one word of 0 of integer type:

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

Inserted in B:

```
B169                                     if prec = 0
B170                                     then goto no_digits; /* no count, as in
    "dec a" */
```

B171

Preceding:

```
A169                                     call assign_ (addr (x (1)), 2 *
    real_fix_bin_2_dtype, 71, addr (buffer),
```

Inserted in B:

```
B225                                     if prec = 0
B226                                     then goto no_digits;
B227
```

Preceding:

```
A222                                     if ^attr.binary & ^attr.float
```

Inserted in B:

```
B285      /* The field has no digits. decevl_ stops at the first character it cannot
B286      use, so this happens for an operand that is not a number at all ("dec foo",
B287      where each letter is taken as an exponent letter, or "dec *+1", where the
B288      field ends at the "*"), for an empty operand, and for a sign or point with
B289      no digits. assign_ cannot convert a number of precision 0 (it takes an
B290      illegal_procedure fault), so flag the field as malformed (E) and return 0,
B291      as for a size condition. The rest of the operand is not skipped, so it
B292      may cause further errors on the same line. */
```

B293

```
B294      no_digits:
```

B295

Preceding:

```
A279      /* A size condition occured. This was probably in assign_, but could have
```

```
Comparison finished: 3 differences, 17 lines.
```

A comment at `no.digits` explains the cases and why `assign_` must not be called. `decevl_` gives the same result in both passes, so the location counter stays consistent (no P error). As with a size condition, the rest of the operand is not skipped, so further errors may be reported on the same line; for `dec **1`, `pass1_` also rejects the `*` that follows the field.

Valid fields always contain at least one digit, so their results are unchanged.

2.1 Version

ALM's version string (`alm.pl1`: "ALM Version 8.14 March 1989") is not changed by the above `compare_ascii` output.

However, after all the testing is done (see below), we should also update the version number:

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

```
A97          global_info.gen_version = "ALM Version 8.14 March 1989";
```

```
Changed by B to:
```

```
B97          global_info.gen_version = "ALM Version 8.15 October 2026";
```

```
Comparison finished: 1 difference, 2 lines.
```

The version string is recorded in every object segment ALM produces. Because of this, it will be easier to perform a comparison of objects produced with the old and new assemblers, if we don't change the version number. However, after we prove that the new assembler assembles the same objects as the old, we should update the version number to 8.15.

3 Documentation

No documentation changes; the E flag is already documented in `alm.info`.

4 Test

1. **The reported cases.** Assemble the two programs from the ticket with the new ALM. Each produces an E error on the `dec` line, the "Translation failed" message, and no fault.
2. **Other fields without digits.** Assemble one program per line below (each with an `end`); each gives an E error and no fault:

```
dec
dec  -
dec  .
dec  a
dec  .e5
dec_unal      foo
```

```

lda    =+
lda    =.
lda    =+aABC

```

3. **Valid fields unchanged.** Assemble a program using the decimal forms (dec 5, dec -5,+7, dec 5L, dec 1.5, dec 1.5d0, dec 1e5, dec 5b17, dec_unal 3, lda =5, lda =-1.5, lda =4aABCD, lda =5,d1) with the old and new ALM; `compare_object` shows identical text.
4. **System sources.** Assemble a sample of system ALM sources (for example those in `bound_alm_` itself and in `bound_library_wired_`) with the new ALM and compare the objects with those from the old ALM: the text is identical.

5 Files Changed

<code>bound_alm_.s.archive</code>	<code>decevl_.pl1</code> : no assign_ call for a field without digits
<code>bound_alm_.s.archive</code>	<code>alm.pl1</code> : update version number
<code>bound_alm_.archive</code>	<code>decevl_</code> (compiled)
<code>bound_alm_.archive</code>	<code>alm</code> (compiled)
<code>>sss</code>	<code>bound_alm_</code> (rebound)

6 Version History

2026-10-10	1.0	Initial version of the MCR.
------------	-----	-----------------------------