MCR10209
display_ttt -of Output Rejected by cv_ttf
Revision 1.0

Eric Swenson

November 22, 2026

1 Problem Description

Ticket #221 describes an issue where the `display_ttt` command, when invoked with the `-of PATH` control argument, fails to generate a TTF segment that can be converted back to a `TTT` using the `cv_ttf` command. This control argument is intended to produce a TTF segment that will translate back to the same table. `cv_ttf`, however, rejects the output:

```
FATAL ERROR 37, SEVERITY 3 IN LINE 2868
Repetition factors nested too deep.
SOURCE:
          initial_string: (111077) <H ETX ESC "A11" ESC "‘;" (12) <NUL> ESC ...
```

The `(111077) <` is not in the original TTF.

1.1 Cause

A string defined with replication factors (such as an `initial_string`) is stored by `cv_ttf` (`store_string`) as the expanded string (`len` characters), then the length of the encoded string as a `fixed bin (17)` occupying two characters, then the encoded string, which therefore begins at character `len + 3`. (Two-character delimiters were introduced in 1988, history entry 7 of `display_ttt`.)

`display_ttt`'s `print_str` reads the length correctly at `source.len + 1`, but starts decoding at `source.len + 2`, the second character of the length. The first "character" decoded is the low half of the length; when it is 256 or more it is taken for a replication code and combined with the next character, giving a bogus factor such as `(111077) <`. That opens an extra group, so the real groups that follow are nested too deeply, and the decoding also ends one character early.

`print_str`'s `count_text`, which groups printable characters, also scanned up to `source.len` (the expanded length) while decoding the encoded string, so it could run past the end of the encoded string.

2 Proposed Fix

In print_str:

```
             stringp = addr (substr (str, source.len + 3));
             stringl, scan_len = code_len;
```

and count_text scans up to the new variable scan_len (source.len for strings without replication factors, code_len for encoded strings). A comment explains the stored format.

The compare_ascii output below shows the proposed changes:

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

Inserted in B:
B942     dcl  scan_len fixed bin;                                  /* length of the part being
    scanned */
Preceding:
A942     dcl  lb bit (1);


Inserted in B:
B968                        scan_len = source.len;
Preceding:
A967                        do i = 1 to source.len;


A984                        stringp = addr (substr (str, source.len + 2));
Changed by B to:
B986
B987     /* cv_ttf stores the expanded string (source.len chars), then the length of
B988        the encoded string in 2 chars, then the encoded string, which therefore
B989        starts at source.len + 3. It was read from source.len + 2, so decoding
B990        started one char early: a bogus first replication factor such as
B991        "(111077) <" made the output unacceptable to cv_ttf (ticket 221). */
B992
B993                        stringp = addr (substr (str, source.len + 3));
B994                        stringl, scan_len = code_len;


A1026                  do j = i to source.len
Changed by B to:
B1036                  do j = i to scan_len

Comparison finished: 4 differences, 14 lines.
```

3 Documentation

No documentation changes.

4 Test

1. `display_ttt -of x.ttf` on the system TTT, then `cv_ttf x`: no errors.
2. Compare the new table with the original, for example with `display_ttt -pn x` against `display_ttt`: identical. The `initial_string`s with replication factors print as in the original TTF source.
3. Terminal types without replication factors are displayed as before.

5 Files Changed

`bound_io_commands_.s.archive`	`display_ttt.pl1`
>sss	`bound_io_commands_` (rebound)

6 Version History

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