

The VFtoVP processor

(Version 1.3, December 2002)

Section	Page
---------	------

1. **Introduction.** The VFtoVP utility program converts a virtual font (\VF

5. Here are some macros for common programming idioms.

de ne *incr*(#

9. The character packets are followed by a trivial postamble, consisting of one or more bytes all equal to *post* (248). The total number of bytes in the file should be a multiple of 4.

χ^2 VFtoVP

13. The rest of the TFM

19. The final portion of a TFM file is the *param* array, which is another sequence of *x_type;1(am)JTJ/F1 9.962625*

28. Of course we want to define macros that suppress the detail of how the font information is actually

32. de ne vf_

48. **Basic output subroutines.** Let u be some procedure that will take θ of.

54. A VPL le has nested parentheses, and we want to format the output so that its structure is clear. The *level*

59. *h*Reduce l by one, preserving the invariants 59 l
begin *decr*(l);
if

66. Outputting the TFM info. T_EX checks the information of a TFM file for validity as the file is being read in, so that no further checks will be needed when typesetting is going on. And when it finds something

70.

79. This program does not check to see if the *seven_bit_safe_ ag*

83.


```
92. h Compute the activity array 92 i  
for ai = 0 to nl - 1 do  
  if activity[ai] = accessible then  
    begin
```

```

96.    hOutput step i of the ligature/kern program 96 i
      begin k    lig_step(i);
      if tfm[k] > stop_ag then
          begin if 256 - k] k

```


110. Checking for ligature loops. We have programmed almost everything but the most interesting calculation of all, which has been saved for last as a special treat. T_E

118. Outputting the VF info. The routines we've used for output from the *tfm* array have counterparts for output from *vf*. One difference is that the string outputs from *vf* need to be checked for balanced parentheses. The *string_balance* routine tests the string of length *l* that starts at location *k*.

function *string*

126. Let's look at the simplest cases first, in order to get some experience.

```
de ne four_cases(#)    #;#
```

129. Before we typeset a character we make sure that it exists.

*h*Cases of DVI instructions that can appear in character packets 126 *i* +

```
sixty_four_cases(set_char_0); sixty_four_cases(set_char_0 + 64); four_cases(set1); four_cases(put1): begin if
  o set1 then
    if o put1 then c get_bytes(o put1 + 1; false)
    else c get_bytes(o set1 + 1; false)
  else c o;
  if f = font_
```


*h*Read and store a font definition [35](#) *i* Used in section [33](#).
h