

adforⁿ*

Clea F. Rees[†]

24th September, 2019

Abstract

Hirwen Harendal, Arkandis Digital Foundry (ADF) has produced Ornaments ADF. This guide outlines the T_EX/L^AT_EX support provided with version 1.001 of the font in postscript type 1 format.

§1 Introduction

This document explains how to use the T_EX/L^AT_EX support included with version 1.001 of Ornaments ADF in postscript type 1 format. The font was developed by Hirwen Harendal of the Arkandis Digital Foundry (ADF), and information about the font itself, together with a copy of the font in opentype format, can be found at <http://pagesperso-orange.fr/arkandis/ADF/tugfonts.htm>. The font is released under the GPL. For details, see README, NOTICE and COPYING.

The T_EX/L^AT_EX support package consists of all files listed in `manifest.txt` and these files are released under the L^AT_EX Project Public Licence as explained in the included licensing notices and README. Please let me know of any problems so that I can solve them if I can. If you can correct the problems and send me the fix, that would be even better. Unlike the font itself, the T_EX/L^AT_EX support is somewhat experimental.

`adforn` includes a copy of the font in type 1 format (`OrnamentsADF.pfb`, `OrnamentsADF.pfm` and `OrnamentsADF.afm`), documentation and support files for T_EX/L^AT_EX including a L^AT_EX package file, `adforn.sty`.

§2 The support package

`adforn` provides access to the ornaments and symbols in `OrnamentsADF` via two sets of commands. First, it provides a single command which takes a range of arguments. The different arguments determine which ornament is typeset. Second, it provides a separate command for each ornament. The choice of command determines which ornament is typeset. The two mechanisms are equivalent¹.

*Version 1.1

[†]ReesC21 <at> cardiff <dot> ac <dot> uk

¹The only difference is that the first allows you to typeset a space by passing it the argument 0 whereas there is no command to typeset the space in the second set. For all practical purposes, this difference is irrelevant since you should not use such a command to typeset a space in T_EX in any case and it is difficult to see why anybody would want to.

§2.1 One command; many arguments

`adorn` provides the command `\adorn{}` which takes a single numerical argument. There are 75 ornaments in the font which can be produced by feeding the relevant number between 1 and 75 to `\adorn{}`²:

1: ◀	16: ≡	31: ♣	46: ∟	61: ♣
2: ▶	17: ∟	32: ♣	47: ∟	62: ♣
3: *	18: ∟	33: ♣	48: ∟	63: ♣
4: *	19: ≡	34: ♣	49: ∟	64: ♣
5: *	20: ∟	35: ♣	50: ∟	65: ♣
6: *	21: ≡	36: ♣	51: ∟	66: ♣
7: *	22: ∟	37: ♣	52: ∟	67: ♣
8: *	23: ∟	38: ♣	53: ∟	68: ♣
9: *	24: ∟	39: ♣	54: ∟	69: ♣
10: *	25: ∟	40: ♣	55: ♣	70: ◀
11: *	26: ≡	41: ♣	56: ♣	71: □
12: *	27: ♣	42: ◀	57: ♣	72: ▶
13: ♣	28: ♣	43: ▶	58: ♣	73: •
14: ◊	29: ♣	44: ∟	59: ♣	74: §
15: ♣	30: ♣	45: ∟	60: ♣	75: ♣

For example,

```
\adorn{21}\quad\adorn{11}\quad\adorn{49}
```

produces:









































































§2.2 Many commands; no arguments

In addition to the numerical interface, a number of additional commands are provided as an alternative means of accessing the various symbols and ornaments. The following list groups them roughly according to kind. In each case, the number of the ornament is given first. This may be used directly with the `\adorn{}` command as explained above. The alternative command is given next. This command may be used to typeset the same ornament. For example both `\adorn{14}` and `\adfdiamond` produce ◊. Finally, the ornament produced by the two commands is typeset to their right.

BASIC SYMBOLS & SHAPES					
74	<code>\adfS</code>	§	75	<code>\adfgee</code>	§
14	<code>\adfdiamond</code>	◊	71	<code>\adfsquare</code>	□
73	<code>\adfbullet</code>	•			

²As mentioned above, the argument 0 will simply typeset a space and should be avoided as using it may interfere with TeX's spacing algorithms. The problem is that TeX will not recognise it as a space and so will treat it instead as a character.

FANCY ASTERISKS & BULLETS					
3	<code>\adfast1</code>		4	<code>\adfast2</code>	
5	<code>\adfast3</code>		6	<code>\adfast4</code>	
7	<code>\adfast5</code>		8	<code>\adfast6</code>	
9	<code>\adfast7</code>		10	<code>\adfast8</code>	
11	<code>\adfast9</code>		12	<code>\adfast{10}</code>	
ARROWS & ARROWHEADS					
70	<code>\adfhalfleftarrow</code>		72	<code>\adfhalfrightarrow</code>	
42	<code>\adfleftarrowhead</code>		43	<code>\adfrightrightarrowhead</code>	
1	<code>\adfhalfleftarrowhead</code>		2	<code>\adfhalfrightarrowhead</code>	
FLOURISHES					
20	<code>\adfflourishleft</code>		48	<code>\adfflourishright</code>	
21	<code>\adfflourishlefthdouble</code>		49	<code>\adfflourishrightdouble</code>	
17	<code>\adfopenflourishleft</code>		45	<code>\adfopenflourishright</code>	
18	<code>\adfclosedflourishleft</code>		46	<code>\adfclosedflourishright</code>	
22	<code>\adfsingleflourishleft</code>		50	<code>\adfsingleflourishright</code>	
19	<code>\adfdoubleflourishleft</code>		47	<code>\adfdoubleflourishright</code>	
26	<code>\adftripleflourishleft</code>		54	<code>\adftripleflourishright</code>	
23	<code>\adfsharpflourishleft</code>		51	<code>\adfsharpflourishright</code>	
24	<code>\adfdoublesharpflourishleft</code>		52	<code>\adfdoublesharpflourishright</code>	
25	<code>\adfsickleflourishleft</code>		53	<code>\adfsickleflourishright</code>	
16	<code>\adfwavesleft</code>		44	<code>\adfwavesright</code>	
FLOWERS					
60	<code>\adfflowerleft</code>		32	<code>\adfflowerright</code>	
LEAVES					
66	<code>\adfleafleft</code>		38	<code>\adfleafright</code>	
59	<code>\adfsolidleafleft</code>		31	<code>\adfsolidleafright</code>	
13	<code>\adfhalfleafleft</code>		15	<code>\adfhalfleafright</code>	
58	<code>\adfoutlineleafleft</code>		30	<code>\adfoutlineleafright</code>	
68	<code>\adfsmallleafleft</code>		40	<code>\adfsmallleafright</code>	
64	<code>\adfflatleafleft</code>		36	<code>\adfflatleafright</code>	
57	<code>\adfflatleafoutlineleft</code>		29	<code>\adfflatleafoutlineright</code>	
65	<code>\adfflatleafsolidleft</code>		37	<code>\adfflatleafsolidright</code>	
67	<code>\adfdownleafleft</code>		39	<code>\adfdownleafright</code>	
61	<code>\adfdownhalfleafleft</code>		33	<code>\adfdownhalfleafright</code>	
55	<code>\adfflatdownhalfleafleft</code>		27	<code>\adfflatdownhalfleafright</code>	
56	<code>\adfflatdownoutlineleafleft</code>		28	<code>\adfflatdownoutlineleafright</code>	
35	<code>\adfhangingleafleft</code>		63	<code>\adfhangingleafright</code>	
69	<code>\adfsmallhangingleafleft</code>		41	<code>\adfsmallhangingleafright</code>	
62	<code>\adfhangingleafleft</code>		34	<code>\adfhangingleafright</code>	

So,

`\adfflourishleftdouble\quad\adfast9\quad\adfflourishrightdouble`

will produce the same output as the example code given in the previous section:

~ * ~