



```

LL      000000  HH      HH      000000  RRRRRRRR  IIIIII
LL      000000  HH      HH      000000  RRRRRRRR  IIIIII
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HHHHHHHHHH  00      00  RRRRRRRR  II
LL      00      00  HHHHHHHHHH  00      00  RRRRRRRR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LL      00      00  HH      HH      00      00  RR      RR  II
LLLLLLLL 000000  HH      HH      000000  RR      RR  IIIIII
LLLLLLLL 000000  HH      HH      000000  RR      RR  IIIIII

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLL IIIIII  SSSSSSSS
LLLLLLLL IIIIII  SSSSSSSS

```

.....

```

1 0001 0 %TITLE 'Line output (horizontal motion)'
2 0002 0 MODULE LOHORI ( IDENT = 'V04-000'
3 P 0003 0 %BLISS32[, ADDRESSING_MODE(EXTERNAL = LONG_RELATIVE,
4 0004 0 NONEXTERNAL = LONG_RELATIVE)]
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
12 0012 1 * ALL RIGHTS RESERVED. *
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
19 0019 1 * TRANSFERRED. *
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
23 0023 1 * CORPORATION. *
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 **
32 0032 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS
33 0033 1
34 0034 1 ABSTRACT: Translation from intermediate format to final output.
35 0035 1
36 0036 1
37 0037 1 ENVIRONMENT: Transportable
38 0038 1
39 0039 1 AUTHOR: K. A. Dawson CREATION DATE: December 1983
40 0040 1

```

```
: 42 0041 1 XSBTTL 'Revision History'  
: 43 0042 1  
: 44 0043 1 MODIFIED BY:  
: 45 0044 1  
: 46 0045 1 010 KFA00010 Ken Alden 06-Jul-1983  
: 47 0046 1 Fixed logic for resetting pointer that is used in  
: 48 0047 1 scanning the MRA during a no-out run-through.  
: 49 0048 1  
: 50 0049 1 009 KFA00009 Ken Alden 30-Jun-1983  
: 51 0050 1 Adding a tsf_cref_count to keep track of pending crefs.  
: 52 0051 1  
: 53 0052 1 008 KFA00008 Ken Alden 28-Jun-1983  
: 54 0053 1 Fixed cref bug with bars enabled.  
: 55 0054 1  
: 56 0055 1 007 KFA00007 Ken Alden 28-Jun-1983  
: 57 0056 1 CLH is not called now unless line has something in it.  
: 58 0057 1  
: 59 0058 1 006 KFA00006 Ken Alden 27-Jun-1983  
: 60 0059 1 Teaked the counter logic for skip_out output.  
: 61 0060 1  
: 62 0061 1 005 KFA00005 Ken Alden 23-Jun-1983  
: 63 0062 1 Added call to OUTCREF when gca_skip_out was true (and a cref  
: 64 0063 1 is pending. This insures that the when the mra is not read,  
: 65 0064 1 OUTCREF may still be called.  
: 66 0065 1  
: 67 0066 1 004 REM00004 Ray Marshall 17-June-1983  
: 68 0067 1 Add call to OUTCREF based on encountering a <RINTES>C escape  
: 69 0068 1 sequence in the MRA.  
: 70 0069 1  
: 71 0070 1 003 KAD00003 Keith Dawson 4-May-1983  
: 72 0071 1 Move test for /QUICK, so that CLH is never called if we are  
: 73 0072 1 skipping output.  
: 74 0073 1  
: 75 0074 1 002 KAD00002 Keith Dawson 14-Apr-1983  
: 76 0075 1 Correct DSRPLUS conditionals for emphasis routines.  
: 77 0076 1  
: 78 0077 1 001 KAD00001 Keith Dawson 22-Mar-1983  
: 79 0078 1  
: 80 0079 1 --
```

```

82 0080 1 %SBTTL 'Module Level Declarations'
83 0081 1
84 0082 1
85 0083 1  TABLE OF CONTENTS:
86 0084 1
87 0085 1
88 0086 1  REQUIRE 'REQ:RNODEF';           ! RUNOFF variant definitions
89 0217 1
90 0218 1  FORWARD ROUTINE
91 0219 1      LOUT1 : NOVALUE,
92 0220 1      build_line,
93 0221 1      compute_next_pass;
94 0222 1
95 0223 1  INCLUDE FILES:
96 0224 1
97 0225 1  LIBRARY 'NXPORT:XPORT';       ! XPORT Library
98 0226 1
99 0227 1  %IF DSRPLUS %THEN
100 U 0228 1  LIBRARY 'REQ:DPLLIB';         ! DSRPLUS BLISS Library
101 U 0229 1  %ELSE
102 0230 1  LIBRARY 'REQ:DSRLIB';       ! DSR BLISS Library
103 0231 1  %FI
104 0232 1
105 0233 1  MACROS:
106 0234 1
107 0235 1
108 0236 1  MACRO
109 M 0237 1      emphasis_passes =
110 M 0238 1      ( (.pass_cntr GTR pass_setup)
111 M 0239 1        AND (.pass_cntr LSS pass_real_text)
112 M 0240 1      )
113 0241 1      %;
114 0242 1
115 0243 1  MACRO
116 M 0244 1      doing_underlining =
117 M 0245 1      (.pass_cntr EQL pass_underline
118 M 0246 1      OR
119 M 0247 1      .pass_cntr EQL pass_bold_underline)
120 0248 1      %;
121 0249 1
122 0250 1  MACRO                               ! TRUE if output is for an
123 M 0251 1      laser_output =                 ! LN01 or an LN01E.
124 M 0252 1      (.gca_op_dev EQL op_dev_ln01
125 M 0253 1      OR
126 M 0254 1      .gca_op_dev EQL op_dev_ln01e)
127 0255 1      %;
128 0256 1
129 0257 1  MACRO                               ! TRUE if we should generate
130 M 0258 1      generate_bare_cr_line =           ! intermediate output.
131 M 0259 1      (
132 M 0260 1      ( (.last_pass GTR 0)           ! First BUILD_LINE loop...
133 M 0261 1      AND                          ! and either we are doing bolding/overstriking,
134 M 0262 1      ( (NOT doing_underlining)     ! or we are doing underlining but not /SEPERATE.
135 M 0263 1      OR
136 M 0264 1      (doing_underlining AND NOT .outopt_und_sep)
137 M 0265 1      )
138 M 0266 1      )

```

```

139 M 0267 1 OR
140 M 0268 1 ( (.last_pass LSS 0)
141 M 0269 1 AND
142 M 0270 1 (.tsf_bld)
143 M 0271 1 )
144 M 0272 1 )
145 M 0273 1 x;
146 M 0274 1
147 M 0275 1 EQUATED SYMBOLS:
148 M 0276 1
149 M 0277 1
150 M 0278 1 EXTERNAL LITERAL
151 M 0279 1 rintes : UNSIGNED (8);
152 M 0280 1
153 M 0281 1
154 M 0282 1 OWN STORAGE:
155 M 0283 1
156 M 0284 1 OWN
157 M 0285 1 emphasis_bits,
158 M 0286 1
159 M 0287 1 p_lines,
160 M 0288 1 padding : VECTOR [75],
161 M 0289 1 pi,
162 M 0290 1 overstrike_count,
163 M 0291 1 overstrike_char,
164 M 0292 1 overstrike_seq,
165 M 0293 1 bold_limit,
166 M 0294 1 over_limit,
167 M 0295 1 under_limit,
168 M 0296 1 pass_limit,
169 M 0297 1 next_pass,
170 M 0298 1 last_pass,
171 M 0299 1 pass_cntr;
172 M 0300 1
173 M 0301 1
174 M 0302 1 EXTERNAL REFERENCES:
175 M 0303 1
176 M 0304 1
177 M 0305 1 EXTERNAL
178 M 0306 1 fnct : FNCT_DEFINITION,
179 U 0307 1 %IF FLIP %THEN
180 U 0308 1 rnoiob: REF $XPO_IOB(),
181 U 0309 1 %FI
182 M 0310 1
183 M 0311 1 fra : fixed_string,
184 M 0312 1 gca : gca_definition,
185 M 0313 1 sca : sca_definition,
186 M 0314 1 tsf : tsf_definition,
187 M 0315 1 outopt : VECTOR [outopt_size],
188 M 0316 1 phan : phan_definition;
189 M 0317 1
190 M 0318 1 EXTERNAL LITERAL
191 M 0319 1 rnfile;
192 M 0320 1
193 M 0321 1 EXTERNAL ROUTINE
194 M 0322 1 usemph, opemph,
195 M 0323 1

```

```

! Second BUILD_LINE loop...
! and any bolding is present.

```

```

! A word containing information on current-character
! and last-character bold and underline.
! Number of physical lines represented.
! Justification spacing built up here.
! Index into padding.
! Number of characters in an overstrike sequence (/BACKSPACE mode only).
! The character with which to overstrike the previous one.
! CH$PTR to start of an overstrike sequence.
! Location of last character to be bolded.
! Location of last overstruck character.
! Location of last character to be underlined.
! Limit of scan for current pass
! Keeps track of the next pass.
! The number of the last pass for this line.
! Count of which pass is happening. See below.

```

```

! Error messages

```

```
: 196      0324 1 %IF LN01 %THEN  
: 197      0325 1      lnemph : NOVALUE,  
: 198      0326 1 %FI  
: 199      U 0327 1 %IF DSRPLUS %THEN  
: 200      U 0328 1      vtemph : NOVALUE,   outcref,  
: 201      U 0329 1 %FI  
: 202      U 0330 1 %IF FLIP %THEN  
: 203      U 0331 1      flemph : NOVALUE,  
: 204      0332 1 %FI  
: 205      0333 1  
: 206      0334 1      clh,      cskipl,      erms,      fbwait,  
: 207      0335 1      justf,    lstops,    newpag,    tpfeql,  
: 208      0336 1      tpr,      uform;
```

```

210 0337 1 %SBTTL 'LOUT1 -- Process remaining normal text in line.'
211 0338 1 GLOBAL ROUTINE LOUT1 (ptr) : NOVALUE =
212 0339 1
213 0340 1 |++
214 0341 1 | FUNCTIONAL DESCRIPTION:
215 0342 1 |
216 0343 1 |     Process the remaining normal text in the line.
217 0344 1 |
218 0345 1 | FORMAL PARAMETERS:
219 0346 1 |
220 0347 1 |     ptr             Character reader in input line.
221 0348 1 |
222 0349 1 | IMPLICIT INPUTS:   None
223 0350 1 |
224 0351 1 | IMPLICIT OUTPUTS: None
225 0352 1 |
226 0353 1 | ROUTINE VALUE:
227 0354 1 | COMPLETION CODES: None
228 0355 1 |
229 0356 1 | SIDE EFFECTS:     None
230 0357 1 |
231 0358 1 | --
232 0359 1 |
233 0360 2 | BEGIN
234 0361 2 |
235 0362 2 | LOCAL
236 0363 2 |     status;
237 0364 2 |
238 0365 2 |     pass_cntr = 1;
239 0366 2 |     bold_limit = 0;
240 0367 2 |     over_limit = 0;
241 0368 2 |     under_limit = 0;
242 0369 2 |     status = false;
243 0370 2 |
244 0371 2 |
245 0372 2 | INCR i FROM 0 TO 74 DO padding [.i] = 1;
246 0373 2 |
247 0374 2 | ! Compute number of physical lines that this record represents.
248 0375 2 | p_lines = 1;
249 0376 2 |
250 0377 2 | !F (.tsf_und
251 0378 3 |     AND .outopt_und_sep)
252 0379 2 | THEN
253 0380 2 |     p_lines = 2;           ! Underline with dashes on next line.
254 0381 2 |
255 0382 2 | ! This 'turns a page' if necessary.
256 0383 2 | IF NOT .fnct_expanding
257 0384 2 | THEN
258 0385 2 |     BEGIN
259 0386 3 |         IF NOT tpr (.p_lines)
260 0387 3 |         THEN
261 0388 3 |             phan_top_page = .phan_paging OR .phan_top_page;
262 0389 3 |
263 0390 3 |     ! If we are positioned at precisely the position where it would be ok
264 0391 3 |     ! to output one or more footnotes, terminate a new page.
265 0392 4 |     IF (tpfeql () NEQ 0)
266 0393 3 |     THEN

```



```

267 0394 3          phan_top_page = .phan_paging OR .phan_top_page;
268 0395          END;
269 0396
270 0397          ! If necessary, put heading on page before writing text.
271 0398          IF (.phan_top_page
272 0399             AND NOT .fnct_expanding)
273 0400          THEN
274 0401             newpag ();
275 0402
276 0403          ! If skipping output because the user used the /PAGES switch (or because
277 0404          ! we are in an early pass of /AUTOMATIC processing), just count the lines
278 0405          ! but don't do any output.
279 0406          IF .gca_skip_out
280 0407          THEN
281 0408             BEGIN
282 0409             %IF DSRPLUS %THEN
283 0410                 LOCAL
284 0411                 temp_ptr,
285 0412                 temp_length;
286 0413             %FI
287 0414
288 0415             phan_lines_tp = .phan_lines_tp + .p_lines;
289 0416             %IF DSRPLUS %THEN
290 0417                 temp_length = .tsf_int_hl;
291 0418                 temp_ptr = .ptr;
292 0419                 WHILE (.tsf_cref_data NEQ 0)
293 0420                     AND
294 0421                     (.temp_length NEQ 0)
295 0422                     AND
296 0423                     (NOT CH$FAIL (.temp_ptr))          DO
297 0424                     BEGIN
298 0425                         temp_ptr = CH$FIND SUB (.temp_length, .temp_ptr, 3
299 0426                         , CH$PTR(OPLIT(%STRING(%CHAR (28), %C ' ')));
300 0427                     IF NOT CH$FAIL (.temp_ptr)
301 0428                     THEN
302 0429                         BEGIN
303 0430                             outcref ();          !Dump this pending cref.
304 0431                             tsf_cref_count = .tsf_cref_count - 1; ! One less cref pending.
305 0432                             !Reduce the context length to reflect what it has already scanned.
306 0433                             temp_length = .tsf_int_hl - CH$DIFF(.temp_ptr, .ptr) - 3;
307 0434                             temp_ptr = CH$PLUS(.temp_ptr, 3);
308 0435                         END;
309 0436                     END;
310 0437             %FI
311 0438             RETURN
312 0439             END;
313 0440
314 0441          ! Compute spacing for justification.
315 0442          IF .tsf_jus_cnt NEQ 0          ! If justification required,
316 0443          THEN                          ! set up PADDING accordingly.
317 0444             justf (padding, .tsf_jus_cnt,
318 0445                    (IF .tsf_justify THEN .tsf_padding ELSE 0),
319 0446                    .tsf_just_alg);
320 0447
321 0448          ! Take care of possible pending formfeed.
322 0449          IF .phan_form_pend NEQ 0
323 0450          THEN

```

```
324 0451 2 IF .phan_simulate
325 0452 2 THEN
326 0453 2 uform ( ) !/SIMULATE
327 0454 2 ELSE
328 0455 2 IF .phan_pause
329 0456 2 THEN
330 0457 2 fbwait ( )
331 0458 2 ELSE
332 U 0459 2 %IF FLIP %THEN
333 UU 0460 2 IF NOT (.gca_op_dev EQL op_dev_flip)
334 U 0461 2 THEN
335 0462 2 %FI
336 0463 2 ! We must write out the formfeed here and then clear the
337 0464 2 ! FRA, because if an emphasized title is waiting, the FRA
338 0465 2 ! is going to be cleared (in BUILD_LINE) before it has
339 0466 2 ! a chance to be written.
340 0467 2 BEGIN
341 0468 2 fs wchar (fra, .phan_form_pend);
342 0469 2 clh (clh_out nocrlf);
343 0470 2 fs init (fra);
344 0471 2 END;
345 0472 2
346 0473 2 phan_form_pend = 0;
347 0474 2
348 0475 2 +
349 0476 2 ! Generation of what TSF/MRA represent happens below this point.
350 0477 2
351 0478 2 ! Take care of actual line printing, including bold, overstriking,
352 0479 2 ! and underlining. This, if not done using backspace, requires several
353 0480 2 ! passes over the line to generate separate lines which can then
354 0481 2 ! be used to overstrike each other.
355 0482 2 -
356 0483 2 ! Make sure the pass counter is 1 going into BUILD_LINE. This avoids a
357 0484 2 ! nasty bug involving recursive calls to LOUT for fop-of-page processing.
358 0485 2
359 0486 2 pass cntr = 1;
360 0487 2 WHILE NOT .status DO
361 0488 2 status = build_line (.ptr);
362 0489 2
363 0490 2 !Processing continues here when we exit BUILD_LINE returning TRUE.
364 0491 2 !1. This output statement (also) does the last overprint
365 0492 2 ! to achieve the proper bolding depth.
366 0493 2 !2. In /BACKSPACE mode, this write statement does the
367 0494 2 ! actual output, since nothing has been output yet.
368 0495 2 !3. In either case, the terminating CRLF is output.
369 U 0496 2 %IF FLIP %THEN
370 UU 0497 2 IF (.gca_op_dev EQL op_dev_flip) and .sca_header
371 UU 0498 2 THEN
372 UU 0499 2 BEGIN
373 UU 0500 2 OWN tochl_rec : $flip_tochl PRESET ([tochl_code] = flip$k_tochl);
374 UU 0501 2 $XPO PUT (IOB=.rnoiob, STRING= (flip$k_tochl_size,tochl_rec));
375 UU 0502 2 sca_header = false;
376 U 0503 2 END;
377 0504 2 %FI
378 0505 2 op_dev_write_output_line;
379 0506 2
380 0507 2 ! Clear output buffer for next line.
```

```

: 381 0508 2 fs_init (fra);
: 382 0509 2
: 383 0510 2 !+ Generate separate underlining now, if specified.
: 384 0511 2 !-
: 385 0512 2 IF .outopt_und_sep
: 386 0513 2 AND
: 387 0514 2 .tsf_und !Not unless there is any underlining to do!
: 388 0515 2 THEN
: 389 0516 2 BEGIN
: 390 0517 4 pass_cntr = (IF .tsf_bld
: 391 0518 4 THEN pass_bold_underline
: 392 0519 4 ELSE pass_underline);
: 393 0520 4 last_pass = -1; ! "-1" signals this separate-underline call to BUILD_LINE.
: 394 0521 4 build_line (.ptr);
: 395 0522 4 fs_wchar (fra, 10); ! Add <lf> to the line.
: 396 0523 4 clh (clh_out_nocrlf);
: 397 0524 4 fs_init (fra); ! Clear output buffer for next line.
: 398 0525 4 phan_lines_tp = .phan_lines_tp + 1;
: 399 0526 2 END;
: 400 0527 2
: 401 0528 2 ! Update count of number of lines on this page.
: 402 0529 2 phan_lines_tp = .phan_lines_tp + 1;
: 403 0530 2 END;

```

.TITLE LOHORI Line output (horizontal motion)  
.IDENT \V04-000\

.PSECT \$OWNS,NOEXE,2

```

00000 EMPHASIS_BITS:
      .BLKB 4
00004 P_LINES: .BLKB 4
00008 PADDING: .BLKB 300
00134 PI: .BLKB 4
00138 OVERSTRIKE_COUNT:
      .BLKB 4
0013C OVERSTRIKE_CHAR:
      .BLKB 4
00140 OVERSTRIKE_SEQ:
      .BLKB 4
00144 BOLD_LIMIT:
      .BLKB 4
00148 OVER_LIMIT:
      .BLKB 4
0014C UNDER_LIMIT:
      .BLKB 4
00150 PASS_LIMIT:
      .BLKB 4
00154 NEXT_PASS:
      .BLKB 4
00158 LAST_PASS:
      .BLKB 4
0015C PASS_CNTR:
      .BLKB 4

```

.EXTRN RINTES, FNCT, FRA

					.EXTRN	GCA, SCA, TSF, OUTOPT		
					.EXTRN	PHAN, RNFILE, BSEMPH		
					.EXTRN	OPEMPH, LNEMPH, CLH		
					.EXTRN	CSKIPL, ERMS, FBWAIT		
					.EXTRN	JUSTF, LSTOPS, NEWPAG		
					.EXTRN	TPFEQL, TPR, UFORM		
					.PSECT	\$CODE\$,NOWRT,2		
					.ENTRY	LOUT1, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	0338	
					MOVAB	BUILD LINE, R11		
					MOVAB	FNCT+24, R10		
					MOVAB	OUTOPT+8, R9		
					MOVAB	CLH, R8		
					MOVAB	TSF, R7		
					MOVAB	P LINES, R6		
					MOVAB	PHAN, R5		
					MOVAB	FRA+4, R4		
0158					MOVL	#1, PASS CNTR	0365	
					CLRQ	BOLD LIMIT	0366	
					CLRL	UNDER LIMIT	0368	
					CLRL	STATUS	0369	
					CLRL	I	0372	
					MOVL	#1, PADDING[I]		
F3	04	A640			AOBLEQ	#74, I, 1\$		
					MOVL	#1, P LINES	0375	
					MOVL	TSF, R0	0377	
06	08	A0			BBC	#1, 8(R0), 2\$		
					BLBC	OUTOPT+8, 2\$	0378	
					MOVL	#2, P LINES	0380	
					BLBS	FNCT+24, 4\$	0383	
					PUSHL	P LINES	0386	
					CALLS	#T, TPR		
					BLBS	R0, 3\$		
					BISL2	@PHAN+40, PHAN	0388	
					CALLS	#0, TPFEQL	0392	
					TSTL	R0		
					BEQL	4\$		
					BISL2	@PHAN+40, PHAN	0394	
					BLBC	PHAN, 5\$	0398	
					BLBS	FNCT+24, 5\$	0399	
					CALLS	#0, NEWPAG	0401	
					BLBC	GCA+112, 6\$	0406	
					ADDL2	P_LINES, PHAN+12	0415	
					RET		0408	
					MOVL	TSF, R0	0442	
					TSTL	32(R0)		
					BEQL	9\$		
					PUSHL	100(R0)	0446	
					BLBC	36(R0), 7\$	0445	
					PUSHL	64(R0)		
					BRB	8\$		
					CLRL	-(SP)		
					PUSHL	32(R0)	0444	
					PUSHAB	PADDING		
					CALLS	#4, JUSTF		
					MOVL	PHAN+32, R2	0449	

			34	13	000CB	BEQL	12\$					
	09	34	A5	E9	000CD	BLBC	PHAN+52, 10\$		0451			
	EF		00	FB	000D1	CALLS	#0, UFORM		0453			
			27	11	000D8	BRB	12\$					
	09	3C	A5	E9	000DA	10\$:	BLBC	PHAN+60, 11\$	0455			
	EF		00	FB	000DE	CALLS	#0, FBWAIT		0457			
			1A	11	000E5	BRB	12\$					
	00	B4	52	90	000E7	11\$:	MOVB	R2, @FRA+4	0468			
			64	D6	000EB	INCL	FRA+4					
			08	A4	D6	000ED	INCL	FRA+12				
			08	0B	DD	000F0	PUSHL	#11	0469			
	68		01	FB	000F2	CALLS	#1, CLH					
			08	A4	D4	000F5	CLRL	FRA+12	0470			
	FC	A4	0C	A4	9E	000F8	MOVAB	FRA+16, FRA				
		64	FC	A4	D0	000FD	MOVL	FRA, FRA+4				
			20	A5	D4	00101	12\$:	CLRL	PHAN+32			
	0158	C6	01	D0	00104	MOVL	#1, PASS_CNTR		0473			
		08	53	E8	00109	13\$:	BLBS	STATUS, T4\$	0486			
			04	AC	DD	0010C	PUSHL	PTR	0487			
			68	01	FB	0010F	CALLS	#1, BUILD_LINE	0488			
			53	50	D0	00112	MOVL	R0, STATUS				
				F2	11	00115	BRB	13\$				
	02	0000000G	EF	04	04	ED	00117	14\$:	CMPZV	#4, #4, GCA+208, #2	0503	
					04	14	00120	BGTR	15\$			
					06	DD	00122	PUSHL	#6			
					02	11	00124	BRB	16\$			
					08	DD	00126	15\$:	PUSHL	#11		
					68	01	FB	00128	16\$:	CALLS	#1, CLH	
					08	A4	D4	0012B	CLRL	FRA+12	0502	
					FC	A4	9E	0012E	MOVAB	FRA+16, FRA		
					FC	A4	D0	00133	MOVL	FRA, FRA+4		
					44	69	E9	0C137	BLBC	OUTOPT+8, 19\$	0512	
					50	67	D0	0013A	MOVL	TSF, R0	0514	
					3C	08	A0	E1	0C13D	BBC	#1, 8(R0), 19\$	
					50	67	D0	00142	MOVL	TSF, R0	0517	
					05	08	A0	E9	00145	BLBC	8(R0), 17\$	
					50	04	D0	00149	MOVL	#4, R0		
						03	11	0014C	BRB	18\$		
					50	03	D0	0014E	17\$:	MOVL	#3, R0	
	0158	C6	50	D0	00151	18\$:	MOVL	R0, PASS_CNTR				
	0154	C6	01	CE	00156	MNEGL	#1, LAST_PASS		0520			
			04	AC	DD	0015B	PUSHL	PTR	0521			
					68	01	FB	0015E	CALLS	#1, BUILD_LINE		
					00	0A	90	00161	MOVB	#10, @FRA+4	0522	
						64	D6	00165	INCL	FRA+4		
						08	A4	D6	00167	INCL	FRA+12	
						08	0B	DD	0016A	PUSHL	#11	
						68	01	FB	0016C	CALLS	#1, CLH	
						08	A4	D4	0016F	CLRL	FRA+12	
						FC	A4	9E	00172	MOVAB	FRA+16, FRA	
						FC	A4	D0	00177	MOVL	FRA, FRA+4	
						OC	A5	D6	0017B	INCL	PHAN+12	
						OC	A5	D6	0017E	19\$:	INCL	PHAN+12
						04	00181	RET	0529			
									0530			

; Routine Size: 386 bytes, Routine Base: \$CODE\$ + 0000

LOHORI  
V04-000

Line output (horizontal motion)  
LOUT1 -- Process remaining normal text in line.

I 4  
16-Sep-1984 00:51:15  
14-Sep-1984 13:06:57

VAX-11 Bliss-32 V4.0-742  
[RUNOFF.SRC]LOHORI.BLI;1

Page 12  
(4)

LO  
VO

```

405 0531 1 %SBTTL 'BUILD_LINE -- output entire text line, using multiple passes if needed'
406 0532 1 ROUTINE BUILD_LINE (ptr) =
407 0533 1
408 0534 1 !**
409 0535 1 FUNCTIONAL DESCRIPTION:
410 0536 1
411 0537 1     BUILD_LINE does the actual work of building up an output line,
412 0538 1     using multiple passes for overstriking, underlining, and bolding.
413 0539 1
414 0540 1 FORMAL PARAMETERS:
415 0541 1
416 0542 1     ptr             is the input line pointer, passed from LOUT to LOUT1.
417 0543 1
418 0544 1 IMPLICIT INPUTS:
419 0545 1
420 0546 1     Some of the OWN variables of this module (NOUT).
421 0547 1
422 0548 1 IMPLICIT OUTPUTS:     None
423 0549 1
424 0550 1 ROUTINE VALUE:
425 0551 1 COMPLETION CODES:
426 0552 1
427 0553 1     The routine returns TRUE to indicate that more processing is
428 0554 1     required on a line; it returns FALSE to indicate that the line is
429 0555 1     ready for output. It is called in a loop until it returns FALSE.
430 0556 1
431 0557 1 SIDE EFFECTS:
432 0558 1
433 0559 1     Text is written onto FRA.
434 0560 1 --
435 0561 1
436 0562 2 BEGIN
437 0563 2
438 0564 2 LOCAL
439 0565 2     hold_khar,
440 0566 2     hold_operand1,
441 0567 2     hold_seq_start,           ! CH$PTR to start of a character sequence.
442 0568 2     op_code,
443 0569 2     operand1,
444 0570 2     ptr_copy_1;
445 0571 2
446 0572 2 ! Initialize LOCAL variables.
447 0573 2 !
448 0574 2 hold_operand1 = 0;
449 0575 2 operand1 = 0;
450 0576 2 ptr_copy_1 = .ptr;
451 0577 2 next_pass = .pass_cntr;     ! Initialize to the current pass count.
452 0578 2
453 0579 2 ! Initialize OWN variables.
454 0580 2 !
455 0581 2 pi = 0;
456 0582 2
457 0583 2 ! Initialize emphasis and overstriking information for each call.
458 0584 2 !
459 0585 2 emphasis_bits = 0;
460 0586 2 overstrike_count = 0;
461 0587 2 overstrike_char = 0;

```

```

462 0588
463 0589
464 0590
465 0591
466 0592
467 0593
468 0594
469 0595
470 0596
471 0597
472 0598
473 0599
474 0600
475 0601
476 0602
477 0603
478 0604
479 0605
480 0606
481 0607
482 0608
483 0609
484 0610
485 0611
486 0612
487 0613
488 0614
489 0615
490 0616
491 0617
492 0618
493 0619
494 0620
495 0621
496 0622
497 0623
498 0624
499 0625
500 0626
501 0627
502 0628
503 0629
504 0630
505 0631
506 0632
507 0633
508 0634
509 0635
510 0636
511 0637
512 0638
513 0639
514 0640
515 0641
516 0642
517 0643
518 0644

```

```

! For LN01[e] output, overstriking is treated in a special way. In this
! case, bolding/underlining passes produce no text on the FRA; the only
! passes that write to the FRA are pass_overstrike and pass_real_text.
IF (laser_output AND .tsf_ovr)
  AND
  (.pass_cntr LSS pass_overstrike)
  AND
  (.pass_cntr GTR pass_setup)
THEN
  ! Decide which pass comes next and set up the counter for it.
  BEGIN
  pass_cntr = compute_next_pass ();
  RETURN false;
  END;

!+
! For all passes except text-generating one, output spaces instead
! of the change bars.
IF emphasis_passes
THEN
  lstops (lstops_none, false) ! Space over the listing option columns.
ELSE
  lstops (lstops_all, false); ! Output listing options.

! Shift text according to amount computed by .CENTER, etc commands.
INCR i FROM 1 TO .tsf_adjust DO
  fs_wchar (fra, %C^');

! Get limit of scan for this pass.
pass_limit = (CASE .pass_cntr FROM pass_setup TO pass_real_text OF
  SET
    [pass_setup] : .tsf_int_hl; ! 1st pass sets up others.
    [pass_bold] : .bold_limit; ! Last character for bolding
    [pass_overstrike] : .over_limit; ! Last overstriking character
    [pass_bold_overstrike] : .over_limit;
    [pass_underline] : .under_limit; ! Last underlined character
    [pass_bold_underline] : .under_limit;
    [pass_real_text] : .tsf_int_hl; ! 7th pass generates output
  TES);
INCR k FROM 1 TO .pass_limit DO ! Process (horizontal) text.
  BEGIN
  hold_seq_start = .ptr_copy_1; ! Remember start of this sequence.
  hold_khar = CHRCHAR_A (ptr_copy_1);

  IF .hold_khar EQL rintes
  THEN

```



```

519      0645  4      BEGIN
520      0646  4      op_code = CH$RCHAR A (ptr_copy_1);
521      0647  4      hold_operand1 = CH$RCHAR_A (ptr_copy_1);
522      0648  4      k = .k + 2;
523      0649  4
524      0650  4      SELECT .op_code OF
525      0651  4      SET
526      0652  4
527      0653  4      [X'C'B'] :
528      0654  5      (IF .tsf_bld
529      0655  5      THEN
530      0656  5      ! Remember bolding information if bolding wanted.
531      0657  6      BEGIN
532      0658  6      emph_current_bold = true;
533      0659  6      operand1 = .hold_operand1;
534      0660  6      END
535      0661  4      );
536      0662  4
537      U 0663  4 %IF DSRPLUS %THEN
538      UU 0664  4 [X'C'C'] :
539      UU 0665  4 BEGIN
540      UU 0666  4 IF .pass_cntr EQL pass_setup
541      UU 0667  4 THEN
542      UU 0668  4 BEGIN
543      UU 0669  4 outcref(); ! Process pending .REF records.
544      UU 0670  4 tsf_cref_count = tsf_cref_count - 1;
545      UU 0671  4 END;
546      UU 0672  4 operand1 = 0;
547      U 0673  4 END;
548      0674  4 %FI
549      0675  4 [X'C'U'] :
550      0676  5 (IF .tsf_und
551      0677  5 THEN
552      0678  5 ! Remember underlining information if underlining wanted.
553      0679  6 BEGIN
554      0680  6 emph_current_underline = true;
555      0681  6 operand1 = .hold_operand1;
556      0682  6 END
557      0683  4 );
558      0684  4
559      0685  4 [X'C'N'] :
560      0686  4 ! A No-operation
561      0687  5 BEGIN
562      0688  5 0 ! Avoid compiler message
563      0689  4 END;
564      0690  4
565      0691  4 [X'C'I'] :
566      0692  4 ! Insert this character.
567      0693  5 BEGIN
568      0694  5 fs_wchar (fra, .hold_operand1);
569      0695  4 END;
570      0696  4
571      0697  4 [X'C'J'] :
572      0698  4 ! Justification mark
573      0699  5 BEGIN
574      0700  5 ! Insert appropriate amount of spacing here.
575      0701  5

```

```
576 0702 5      INCR i FROM 1 TO .padding [.pi] DO
577 0703 5          fs_wchar (fra, %C' ');
578 0704 5
579 0705 5      operand1 = 0;
580 0706 5      pi = .pi + 1;      ! Synchronize insert count with word
581 0707 4      END;          ! count.
582 0708 4
583 0709 4      [%C'0'] :
584 0710 5      (if .tsf_ovr
585 0711 5      THEN
586 0712 5      ! If overstriking is wanted remember this information.
587 0713 6      BEGIN
588 0714 6      ! Remember overstrike character
589 0715 6      overstrike_char = .hold_operand1;
590 0716 6      operand1 = .hold_operand1;
591 0717 6
592 0718 6      IF .overstrike_count EQL 0
593 0719 6      THEN
594 0720 6      ! Remember start of overstrike sequence.
595 0721 6      overstrike_seq = .hold_seq_start;
596 0722 6      overstrike_count = .overstrike_count + 1;
597 0723 6      END
598 0724 4      );
599 0725 4
600 0726 4      [OTHERWISE] :
601 0727 5      BEGIN
602 0728 5      ! Some illegal character following RINTES. Tell the user
603 0729 5      ! it's an internal logic error and then carry on.
604 0730 5      erms (rnfile, CH$PTR (UPLIT ('lout1')), 5);
605 0731 4      END;
606 0732 4      TES;
607 0733 4      END
608 0734 3      ELSE
609 0735 4      BEGIN      ! Are positioned at the 'naked' character.
610 0736 4      ! Is this an emphasized or overstruck character?
611 0737 4      IF .operand1 NEQ 0
612 0738 4      THEN
613 0739 4      ! Process character according to which pass.
614 0740 5      BEGIN
615 0741 5      IF .pass_cntr EQL pass_setup
616 0742 5      THEN
617 0743 5      ! Save location of emphasized character for later passes.
618 0744 6      BEGIN
619 0745 6      IF .emph_current_bold      THEN bold_limit = .k;
620 0746 6      IF (.overstrike_count NEQ 0) THEN over_limit = .k;
621 0747 6      IF .emph_current_underline THEN under_limit = .k;
622 0748 5      END;
623 0749 5
624 0750 5      SELECT ONE TRUE OF
625 0751 5      SET
626 0752 5      %IF LN01 %THEN
627 0753 5      [laser_output] :      !Output for LN01 or LN01E.
628 0754 5      lnemph (.hold_khar, .gca_ln01_ital_under,
629 0755 5      emphasis_bits, .overstrike_count,
630 0756 5      .overstrike_char, .overstrike_seq, .pass_cntr);
631 0757 5      %FI
632 U 0758 5      %IF DSRPLUS %THEN
```

```

633 U 0759 5 [.gca_op_dev EQL op_dev_vt100] : !Output for VT100.
634 U 0760 5 vtemp (.hold_khar, .gca_ln01_ital_under,
635 U 0761 5 emphasis_bits, .pass_cntr);
636 U 0762 5 %FI
637 U 0763 5 %IF FLIP %THEN
638 U 0764 5 [(.gca_op_dev EQL op_dev_flip)] : !Output for FLIP.
639 U 0765 5 fltemp (.hold_khar, .gca_ln01_ital_under,
640 U 0766 5 emphasis_bits, .pass_cntr);
641 U 0767 5 %FI
642 U 0768 5 [.outopt_back] : !Backspace mode.
643 U 0769 5 btemp (.hold_khar, .gca_ln01_ital_under,
644 U 0770 5 emphasis_bits, .overstrike_count,
645 U 0771 5 .overstrike_char, .overstrike_seq, .pass_cntr);
646 U 0772 5
647 U 0773 5 [.outopt_over] : !Line overprinting mode.
648 U 0774 5 otemp (.hold_khar, .gca_ln01_ital_under,
649 U 0775 5 emphasis_bits, .overstrike_count,
650 U 0776 5 .overstrike_char, .overstrike_seq, .pass_cntr);
651 U 0777 5
652 U 0778 5 [OTHERWISE] :
653 U 0779 5 erms (rfile, CHSPTR (UPLIT ('build_line')), 10);
654 U 0780 5
655 U 0781 5 TES;
656 U 0782 5
657 U 0783 5 operand1 = 0;
658 U 0784 5 hold_operand1 = 0;
659 U 0785 5 emph_current_bold = false;
660 U 0786 5 emph_current_underline = false;
661 U 0787 5 overstrike_count = 0;
662 U 0788 5 overstrike_char = 0;
663 U 0789 5 END
664 U 0790 4 ELSE
665 U 0791 4 !It's a normal, unemphasized character to be output. Put it
666 U 0792 4 ! in the output buffer only if pass 1 or pass 7; otherwise
667 U 0793 4 ! use ' ' for a place holder.
668 U 0794 5 IF (NOT emphasis_passes)
669 U 0795 4 THEN
670 U 0796 5 BEGIN
671 U 0797 5 %IF LN01 %THEN
672 U 0798 5 !Check for LN01 emphasis and turn it off.
673 U 0799 6 IF laser_output
674 U 0800 5 AND
675 U 0801 6 (.emph_previous_emphasized NEQ 0) !Emphasis on?
676 U 0802 5 THEN
677 U 0803 5 !Have to turn off all emphasis
678 U 0804 5 ltemp (-1, .gca_ln01_ital_under,
679 U 0805 5 emphasis_bits, .overstrike_count,
680 U 0806 5 .overstrike_char, .overstrike_seq, .pass_cntr);
681 U 0807 5 %FI
682 U 0808 5 %IF DSRPLUS %THEN
683 U 0809 5 !Check for VT100 emphasis and turn it off.
684 U 0810 5 IF (.gca_op_dev EQL op_dev_vt100) !VT100.
685 U 0811 5 AND
686 U 0812 5 (.emph_previous_emphasized NEQ 0) !Emphasis on?
687 U 0813 5 THEN
688 U 0814 5 !Have to turn off all emphasis
689 U 0815 5 vtemp (-1, .gca_ln01_ital_under,

```

```
690 U 0816 5 emphasis_bits, .pass_cntr);
691 U 0817 5 %FI
692 U 0818 5 %IF FLIP %THEN
693 U 0819 5 !Check for FLIP emphasis and turn it off
694 U 0820 5 IF (.gca_op_dev EQL op_dev_flip)
695 U 0821 5 THEN
696 U 0822 5 flemph (-1, .gca_ln01_ital_under,
697 U 0823 5 emphasis_bits, .pass_cntr);
698 U 0824 5 %FI
699 U 0825 5 fs_wchar (fra, .hold_khar); !First or last pass: write character.
700 U 0826 5 END
701 U 0827 4 ELSE
702 U 0828 4 fs_wchar (FRA, %C' '); ! Emphasis pass: write placeholder space.
703 U 0829 4
704 U 0830 3 END; ! End of 'naked' character processing.
705 U 0831 3
706 U 0832 2 END; ! End of 'INCR K' loop.
707 U 0833 2
708 U 0834 2 ! For the first pass, compute the number of the LAST pass that will be
709 U 0835 2 ! made over this text.
710 U 0836 2
711 U 0837 2 IF .pass_cntr EQL pass_setup
712 U 0838 2 THEN
713 U 0839 2 last_pass =
714 U 0840 4 (IF (.over_limit NEQ 0
715 U 0841 4 OR .under_limit NEQ 0
716 U 0842 4 OR .bold_limit NEQ 0)
717 U 0843 3 THEN
718 U 0844 3 pass_real_text
719 U 0845 3 ELSE
720 U 0846 2 pass_setup);
721 U 0847 2 %IF LN01 %THEN
722 U 0848 2 ! Be sure there's no emphasis left hanging around.
723 U 0849 3 IF (laser_output AND (.emph_previous_emphasized NEQ 0) )
724 U 0850 2 THEN
725 U 0851 2 !Finish this record.
726 U 0852 2 lneiph (-1, .gca_ln01_ital_under,
727 U 0853 2 emphasis_bits, .overstrike_count,
728 U 0854 2 .overstrike_char, .overstrike_seq, .pass_cntr);
729 U 0855 2 %FI
730 U 0856 2 %IF DSRPLUS %THEN
731 U 0857 2 !In VT100 mode everything is done in one pass. But first we have to be
732 U 0858 2 !sure there's no emphasis left hanging around.
733 U 0859 2
734 U 0860 2 IF (.gca_op_dev EQL op_dev_vt100) !VT100?
735 U 0861 2 THEN
736 U 0862 2 !Finish this record
737 U 0863 2 BEGIN
738 U 0864 2 IF .emph_previous_underline OR .emph_previous_bold !Any emphasis left on?
739 U 0865 2 THEN
740 U 0866 2 vtemp (-1, .gca_ln01_ital_under,
741 U 0867 2 emphasis_bits, .pass_cntr);
742 U 0868 2
743 U 0869 2 RETURN true; !Nothing more left to do.
744 U 0870 2 END;
745 U 0871 2 %FI
746 U 0872 2 %IF FLIP %THEN
```

```

: 747      U 0873 2      !For FLIP, everything is done in one pass.
: 748      UU 0874 2      IF (.gca_op_dev EQL op_dev_flip)
: 749      UU 0875 2      THEN
: 750      UU 0876 2      BEGIN
: 751      UU 0877 2      flemph (-1, .gca_ln01_ital_under,
: 752      UU 0878 2      emphasis_bits, .pass_cntr);
: 753      UU 0879 2
: 754      UU 0880 2      RETURN true;
: 755      U 0881 2      END;
: 756      0882 2      XFI
: 757      0883 2
: 758      0884 2      +
: 759      0885 2      We are now finished with one pass ... almost. Regardless of how we got here, there's some text
: 760      0886 2      that hasn't been output yet. If there is nothing left to do, then we'll RETURN TRUE, which forces
: 761      0887 2      out this text with a CRLF after it at the bottom of LOUT1. Otherwise, near the end of this routine
: 762      0888 2      the text gets output with no CRLF.
: 763      0889 2      -
: 764      0890 2      IF (.outopt_back)          ! User said /BACKSPACE; everything is done in one pass.
: 765      0891 2      OR
: 766      0892 2      (.pass_cntr EQL .last_pass)    ! Last pass.
: 767      0893 2      THEN
: 768      0894 2      RETURN true;
: 769      0895 2
: 770      0896 2      ! For emphasis passes, add a "bare <cr>" to the line. It is at the end of the line which is to
: 771      0897 2      ! be overprinted, not at the start of the line which does the overprinting.
: 772      0898 2      IF emphasis_passes
: 773      0899 2      THEN
: 774      0900 2      fs_wchar (fra, 13);
: 775      0901 2      +
: 776      0902 2      The following two output operations, which generate the intermediate (emphasis) bare-<cr>
: 777      0903 2      lines, are not done for the underlining passes if the user specified /SEPARATE. Instead,
: 778      0904 2      underlining is done in a separate call to BUILD_LINE, from LOUT1, after the 7th pass over
: 779      0905 2      the line. This second call to BUILD_LINE is signalled by a LAST_PASS value of -1, which is
: 780      0906 2      used in the macro GENERATE_BARE_CR_LINE, tested below.
: 781      0907 2      -
: 782      0908 2      ! If this is a bolding pass (.PASS_CNTR is even) then repeat the line as many times as specified
: 783      0909 2      ! on the /BOLD:n switch. The expression (.OUTOPT_BLDN - 1) is arrived at as follows: .OUTOPT_BLDN is
: 784      0910 2      ! the number of times that the line should be overprinted, so this INCR overprints one time less. An
: 785      0911 2      ! additional CLH (CLH_OUT_NOCRLF) below adds an overprint. When BUILD_LINE returns TRUE the final
: 786      0912 2      ! overprinting is done. The module DOOPTS has taken care of the /BOLD:0 case, so that if the user said
: 787      0913 2      ! /BOLD:0, no bolding is seen by this routine at all.
: 788      0914 2
: 789      0915 2      IF (NOT .pass_cntr)
: 790      0916 2      AND
: 791      0917 2      generate_bare_cr_line
: 792      0918 2      AND
: 793      0919 2      (NOT laser_output)
: 794      0920 2      THEN
: 795      0921 2      INCR I FROM 1 TO (.outopt_bldn - 1) DO
: 796      0922 2      clh (clh_out_nocrlf);
: 797      0923 2
: 798      0924 2      ! At the end of every intermediate (emphasis) pass over the line, output the line without any
: 799      0925 2      ! carriage control following it. If this is the first or the last pass, then we do no output now.
: 800      0926 2      IF emphasis_passes
: 801      0927 2      AND
: 802      0928 2      generate_bare_cr_line
: 803      0929 2      AND

```

```

804 0930 3 (NOT laser_output)
805 0931 THEN
806 0932 clh (clh_out_nocrlf);
807 0933
808 0934
809 0935 | Clear the line buffer in the following cases:
810 0936 |
811 0937 | - for LN01 output, on any pass but overstriking or real_text
812 0938 | - for non-LN01 output, on any pass but the separate-underlining pass
813 0939
814 0940 IF laser_output
815 0941 THEN
816 0942 | ** NOTE: This test depends on the values in PASS.REQ: the
817 0943 | overstriking passes must come last.
818 0944 BEGIN
819 0945 IF .pass_cntr LSS pass_overstrike
820 0946 THEN
821 0947 fs_init (fra);
822 0948 END
823 0949 ELSE
824 0950 BEGIN
825 0951 IF .last_pass GTR 0
826 0952 THEN !Not separate-underlining.
827 0953 fs_init (fra);
828 0954 END;
829 0955
830 0956 | Decide which pass comes next and set up counter for the next iteration.
831 0957 |
832 0958 pass_cntr = compute_next_pass ();
833 0959
834 0960 RETURN false;
835 0961
836 0962 1 END;

```

! End of build\_line

```

.PSECT $SPLITS$,NOWRT,NOEXE,2
00 00 65 6E 00 00 00 31 74 75 6F 6C 00000 P.AAA: .ASCII \lout1\<0><0><0>
00 00 69 6C 5F 64 6C 69 75 62 00008 P.AAB: .ASCII \build_line\<0><0>

```

```

.PSECT $CODE$,NOWRT,2
OFFC 00000 BUILD_LINE:
5B 00000000' EF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0532
56 7C 00009 MOVAB EMPHASIS_BITS, R11 : 0575
58 04 AC D0 0000B CLRQ OPERAND1 : 0576
51 015C CB D0 0000F MOVL PTR, PTR_COPY_1 : 0577
0154 CB 51 D0 00014 MOVL PASS_CNTR, R1
6B D4 00019 MOVL R1, NEXT_PASS : 0585
0134 CB 7C 0001B CLRL EMPHASIS_BITS : 0581
013C CB D4 0001F CLRL PI : 0587
04 0000000G EF 04 04 ED 00023 CLRL OVERSTRIKE_CHAR : 0593
0B 13 0002C CMPZV #4, #4, GCA+208, #4
BEQL 1$

```

05	00000000G	EF	04	04	ED	0002E		CMPZV	#4, #4, GCA+208, #5			
				19	12	00037		BNEQ	2\$			
			50	00000000G	EF	D0	00039	1\$:	MOVL	TSF, R0		
		OD	08	A0	02	E1	00040		BBC	#2, 8(R0), 2\$		
				05	51	D1	00045		CMPL	R1, #5		
					08	18	00048		BGEQ	2\$		
				01	51	D1	0004A		CMPL	R1, #1		
					03	15	0004D		BLEQ	2\$		
					04	69	0004F		BRW	69\$		
				01	015C	CB	D1	00052	2\$:	CMPL	PASS_CNTR, #1	
					08	15	00057		BLEQ	3\$		
				07	015C	CB	D1	00059		CMPL	PASS_CNTR, #7	
					04	18	0005E		BGEQ	3\$		
					7E	7C	00060		CLRQ	-(SP)		
					03	11	00062		BRB	4\$		
				7E	1F	7D	00064	3\$:	MOVQ	#31, -(SP)		
				00000000G	EF	02	FB	00067	4\$:	CALLS	#2, LSTOPS	
				50	00000000G	EF	D0	0006E		MOVL	TSF, R0	
						51	D4	00075		CLRL	I	
						13	11	00077		BRB	6\$	
				00000000G	FF	20	90	00079	5\$:	MOVB	#32, @FRA+4	
				00000000G	EF	D6	00080		INCL	FRA+4		
				00000000G	EF	D6	00086		INCL	FRA+12		
				E8	51	28	A0	F3	0008C	6\$:	AOBLEQ	40(R0), I, 5\$
				06	01	015C	CB	CF	00091		CASEL	PASS_CNTR, #1, #6
001E				001E	0010	0025			00097	7\$:	.WORD	11\$-7\$, -
				0025	0017	0017			0009F			8\$-7\$, -
												10\$-7\$, -
												10\$-7\$, -
												9\$-7\$, -
												9\$-7\$, -
												11\$-7\$, -
						15	11	000A5		BRB	11\$	0624
				50	0144	CB	D0	000A7	8\$:	MOVL	BOLD_LIMIT, R0	0626
						11	11	000AC		BRB	12\$	
				50	0148	CB	D0	000AE	9\$:	MOVL	OVER_LIMIT, R0	0629
						0A	11	000B3		BRB	12\$	
				50	014C	CB	D0	000B5	10\$:	MOVL	UNDER_LIMIT, R0	0632
						03	11	000BA		BRB	12\$	
				50		60	D0	000BC	11\$:	MOVL	(R0), R0	0634
				0150		50	D0	000BF	12\$:	MOVL	R0, PASS_LIMIT	0621
				59	0150	CB	D0	000C4		MOVL	PASS_LIMIT, R9	0638
						53	D4	000C9		CLRL	K	
						0244	31	000CB		BRW	42\$	
				5A		58	D0	000CE	13\$:	MOVL	PTR_COPY_1, HOLD_SEQ_START	0640
				55		88	9A	000D1		MOVZBL	(PTR_COPY_1)+, HOLD_RHAR	0641
				00000000G	8F	55	D1	000D4		CMPL	HOLD_KHAR, #RINTES	0643
						03	13	000DB		BEQL	14\$	
						00E8	31	000DD		BRW	25\$	
				54		88	9A	000E0	14\$:	MOVZBL	(PTR_COPY_1)+, OP_CODE	0646
				57		88	9A	000E3		MOVZBL	(PTR_COPY_1)+, HOLD_OPERAND1	0647
				53		02	C0	000E6		ADDL2	#2, R	0648
				51		01	D0	000E9		MOVL	#1, R1	0650
				00000042	8F	54	D1	000EC		CMPL	OP_CODE, #66	0653
						13	12	000F3		BNEQ	15\$	
						51	D4	000F5		CLRL	R1	
				50	00000000G	EF	D0	000F7		MOVL	TSF, R0	0654

	06	08	A0	E9	000FE	BLBC	8(R0), 15\$			
	6B		02	88	00102	BISB2	#2, EMPHASIS BITS	0658		
	56		57	D0	00105	MOVL	HOLD_OPERAND1, OPERAND1	0659		
00000055	8F		54	D1	00108	15\$:	CMPL	OP CODE, #85	0675	
			14	12	0010F	BNEQ	16\$			
			51	D4	00111	CLRL	R1			
06	50	00000000G	EF	D0	00113	MOVL	TSF, R0	0676		
	08		A0	E1	0011A	BBC	#1, 8(R0), 16\$			
	6B		04	88	0011F	BISB2	#4, EMPHASIS BITS	0680		
	56		57	D0	00122	MOVL	HOLD_OPERAND1, OPERAND1	0681		
0000004E	8F		54	D1	00125	16\$:	CMPL	OP CODE, #78	0685	
			02	12	0012C	BNEQ	17\$			
			51	D4	0012E	CLRL	R1			
00000049	8F		54	D1	00130	17\$:	CMPL	OP CODE, #73	0691	
			15	12	00137	BNEQ	18\$			
			51	D4	00139	CLRL	R1			
00000000G	FF		57	90	0013B	MOVB	HOLD_OPERAND1, @FRA+4	0694		
		00000000G	EF	D6	00142	INCL	FRA+4			
		00000000G	EF	D6	00148	INCL	FRA+12			
0000004A	8F		54	D1	0014E	18\$:	CMPL	OP CODE, #74	0697	
			28	12	00155	BNEQ	21\$			
	50	0134	CB	D0	00157	MOVL	PI, R0	0702		
			51	7C	0015C	CLRQ	R1	0697		
			13	11	0015E	BRB	20\$	0702		
00000000G	FF		20	90	00160	19\$:	MOVB	#32, @FRA+4	0703	
		00000000G	EF	D6	00167	INCL	FRA+4			
		00000000G	EF	D6	0016D	INCL	FRA+12			
E7	52		08	AB40	F3	00173	20\$:	AOBLEQ	PADDING[R0], I, 19\$	0702
			56	D4	00179	CLRL	OPERAND1	0705		
		0134	CB	D6	0017B	INCL	PI	0706		
0000004F	8F		54	D1	0017F	21\$:	CMPL	OP CODE, #79	0709	
			25	12	00186	BNEQ	23\$			
			51	D4	00188	CLRL	R1			
	50	00000000G	EF	D0	0018A	MOVL	TSF, R0	0710		
17	08		02	E1	00191	BBC	#2, 8(R0), 23\$			
	013C		57	D0	00196	MOVL	HOLD_OPERAND1, OVERSTRIKE_CHAR	0715		
	56		57	D0	0019B	MOVL	HOLD_OPERAND1, OPERAND1	0716		
		0138	CB	D5	0019E	TSTL	OVERSTRIKE_COUNT	0718		
			05	12	001A2	BNEQ	22\$			
	0140		5A	D0	001A4	MOVL	HOLD_SEQ_START, OVERSTRIKE_SEQ	0721		
		0138	CB	D6	001A9	22\$:	INCL	OVERSTRIKE_COUNT	0722	
			15	51	E9	001AD	23\$:	BLBC	R1, 24\$	0726
			05	DD	001B0	PUSHL	#5	0730		
		00000000'	EF	9F	001B2	PUSHAB	P.AAA			
		00000000G	8F	DD	001B8	PUSHL	#RNFILE			
00000000G	EF		03	FB	001BE	CALLS	#3, ERMS			
			014A	31	001C5	24\$:	BRW	42\$	0643	
	51	015C	CB	D0	001C8	25\$:	MOVL	PASS_CNTR, R1	0741	
			56	D5	001CD	TSTL	OPERAND1	0737		
			03	12	001CF	BNEQ	26\$			
			00DD	31	001D1	BRW	36\$			
	01		51	D1	001D4	26\$:	CMPL	R1, #1	0741	
			1D	12	001D7	BNEQ	29\$			
05	08		01	E1	001D9	BBC	#1, EMPHASIS BITS, 27\$	0745		
	0144		53	D0	001DD	MOVL	K, BOLD_LIMIT			
		0138	CB	D5	001E2	27\$:	TSTL	OVERSTRIKE_COUNT	0746	
			05	13	001E6	BEQL	28\$			



05	0148	CB	53	D0	001E8	MOVL	K, OVER LIMIT	
	6B		02	E1	001ED	BBC	#2, EMPHASIS BITS, 29\$	0747
	014C	CB	53	D0	001F1	MOVL	K, UNDER_LIMIT	
04	00000000G	EF	04	D4	001F6	29\$: CLRL	R2	0753
			04	ED	001F8	CMPZV	#4, #4, GCA+208, #4	
			02	12	00201	BNEQ	30\$	
			52	D6	00203	INCL	R2	
05	00000000G	EF	04	D4	00205	30\$: CLRL	R0	
			04	ED	00207	CMPZV	#4, #4, GCA+208, #5	
			02	12	00210	BNEQ	31\$	
			50	D6	00212	INCL	R0	
			52	C8	00214	31\$: BISL2	R2, R0	
			50	D1	00217	CMPL	R0, #1	
			21	12	0021A	BNEQ	32\$	
			51	DD	0021C	PUSHL	R1	0756
		7E	013C	CB	7D	0021E	MOVQ	OVERSTRIKE_CHAR, -(SP)
			0138	CB	DD	00223	PUSHL	OVERSTRIKE_COUNT
				5B	DD	00227	PUSHL	R11
7E	00000000G	EF	01	00	EF	00229	EXTZV	#0, #1, GCA+209, -(SP)
				55	DD	00232	PUSHL	HOLD_KHAR
	00000000G	EF	07	FB	00234	CALLS	#7, [NEMPH	
			01	00000000G	69	11	0023B	BRB
				EF	D1	0023D	32\$: CMPL	OUTOPT+12, #1
				21	12	00244	BNEQ	33\$
				51	DD	00246	PUSHL	R1
		7E	013C	CB	7D	00248	MOVQ	OVERSTRIKE_CHAR, -(SP)
			0138	CB	DD	0024D	PUSHL	OVERSTRIKE_COUNT
				5B	DD	00251	PUSHL	R11
7E	00000000G	EF	01	00	EF	00253	EXTZV	#0, #1, GCA+209, -(SP)
				55	DD	0025C	PUSHL	HOLD_KHAR
	00000000G	EF	07	FB	0025E	CALLS	#7, BSEMPH	
			01	00000000G	3F	11	00265	BRB
				EF	D1	00267	33\$: CMPL	OUTOPT+16, #1
				21	12	0026E	BNEQ	34\$
				51	DD	00270	PUSHL	R1
		7E	013C	CB	7D	00272	MOVQ	OVERSTRIKE_CHAR, -(SP)
			0138	CB	DD	00277	PUSHL	OVERSTRIKE_COUNT
				5B	DD	0027B	PUSHL	R11
7E	00000000G	EF	01	00	EF	0027D	EXTZV	#0, #1, GCA+209, -(SP)
				55	DD	00286	PUSHL	HOLD_KHAR
	00000000G	EF	07	FB	00288	CALLS	#7, OPEMPH	
				15	11	0028F	BRB	35\$
				0A	DD	00291	34\$: PUSHL	#10
				EF	9F	00293	PUSHAB	P.AAB
				8F	DD	00299	PUSHL	#RNFILE
	00000000G	EF	03	FB	0029F	CALLS	#3, ERMS	
			6B	7C	002A6	35\$: CLRQ	OPERAND1	0783
				06	8A	002A8	BICB2	#6, EMPHASIS BITS
				06	7C	002AB	CLRQ	OVERSTRIKE_COUNT
			0138	61	11	002AF	BRB	42\$
				51	D1	002B1	36\$: CMPL	R1, #1
				05	15	002B4	BLEQ	37\$
				51	D1	002B6	CMPL	R1, #7
				44	19	002B9	BLSS	40\$
04	00000000G	EF	04	ED	002BB	37\$: CMPZV	#4, #4, GCA+208, #4	0799
				0B	13	002C4	BEQL	38\$
05	00000000G	EF	04	ED	002C6	CMPZV	#4, #4, GCA+208, #5	

			25	12	002CF		BNEQ	39\$		
		18	6B	93	002D1	38\$:	BITB	EMPHASIS_BITS, #24		0801
			20	13	002D4		BEQL	39\$		
			51	DD	002D6		PUSHL	R1		0806
		7E	013C	CB	7D	002D8	MOVQ	OVERSTRIKE_CHAR, -(SP)		
			0138	CB	DD	002DD	PUSHL	OVERSTRIKE_COUNT		0805
				5B	DD	002E1	PUSHL	R11		0804
7E	00000000G	EF	01	00	EF	002E3	EXTZV	#0, #1, GCA+209, -(SP)		
			7E	01	CE	002EC	MNEGL	#1, -(SP)		
	00000000G	EF	07	FB	002EF		CALLS	#7, LNEMPH		
	00000000G	FF	55	90	002F6	39\$:	MOVB	HOLD_KHAR, @FRA+4		0825
			07	11	002FD		BRB	41\$		
	00000000G	FF	20	90	002FF	40\$:	MOVB	#32, @FRA+4		0828
			00000000G	EF	D6	00306	41\$:	INCL	FRA+4	
			00000000G	EF	D6	0030C		INCL	FRA+12	0825
FDB6		53	01	59	F1	00312	42\$:	ACBL	R9, #1, K, 13\$	0638
			01	1F	12	0031D		CMPL	PASS_CNTR, #1	0837
				015C	CB	D1	00318	BNEQ	46\$	
				1F	12	0031D		TSTL	OVER_LIMIT	0840
				0148	CB	D5	0031F	BNEQ	43\$	
				0C	12	00323		TSTL	UNDER_LIMIT	0841
				014C	CB	D5	00325	BNEQ	43\$	
				06	12	00329		TSTL	BOLD_LIMIT	0842
				0144	CB	D5	0032B	BEQL	44\$	
			50	07	D0	00331	43\$:	MOVL	#7, R0	0840
				03	11	00334		BRB	45\$	
			50	01	D0	00336	44\$:	MOVL	#1, R0	
04	00000000G	EF	0158	04	04	ED	00339	45\$:	MOVL	R0, LAST_PASS
				04	04	ED	0033E	46\$:	CMPZV	#4, #4, GCA+208, #4
05	00000000G	EF		04	04	ED	00347		BEQL	47\$
				04	04	ED	00349		CMPZV	#4, #4, GCA+208, #5
				27	12	00352		BNEQ	48\$	
			18	6B	93	00354	47\$:	BITB	EMPHASIS_BITS, #24	
				22	13	00357		BEQL	48\$	
				015C	CB	DD	00359		PUSHL	PASS_CNTR
			7E	013C	CB	7D	0035D		MOVQ	OVERSTRIKE_CHAR, -(SP)
				0138	CB	DD	00362		PUSHL	OVERSTRIKE_COUNT
				5B	DD	00366		PUSHL	R11	0852
7E	00000000G	EF	01	00	EF	00368		EXTZV	#0, #1, GCA+209, -(SP)	
			7E	01	CE	00371		MNEGL	#1, -(SP)	
	00000000G	EF	07	FB	00374		CALLS	#7, LNEMPH		
			09	00000000G	EF	E8	0037B	48\$:	BLBS	OUTOPT+12, 49\$
			0158	015C	CB	D1	00382		CMPL	PASS_CNTR, LAST_PASS
				04	12	00389		BNEQ	50\$	
			50	01	D0	0038B	49\$:	MOVL	#1, R0	0894
				04	0038E			RET		
			50	015C	CB	D0	0038F	50\$:	MOVL	PASS_CNTR, R0
			01	50	D1	00394		CMPL	R0, #1	0898
				18	15	00397		BLEQ	51\$	
			07	50	D1	00399		CMPL	R0, #7	
				13	18	0039C		BGEQ	51\$	
	00000000G	FF	0D	90	0039E		MOVB	#13, @FRA+4		0900
			00000000G	EF	D6	003A5		INCL	FRA+4	
			00000000G	EF	D6	003AB		INCL	FRA+12	
			63	50	E8	003B1	51\$:	BLBS	R0, 59\$	0915
			51	0158	CB	D0	003B4		MOVL	LAST_PASS, R1
				1F	15	003B9		BLEQ	55\$	0916

			52	D4	0038B	CLRL	R2				
		03	50	D1	0038D	CMPL	RO, #3				
			04	12	003C0	BNEQ	52\$				
			52	D6	003C2	INCL	R2				
			05	11	003C4	BRB	53\$				
		04	50	D1	003C6	CMPL	RO, #4				
			1E	12	003C9	BNEQ	56\$				
		05	52	E8	003CB	BLBS	R2, 54\$				
		04	50	D1	003CE	CMPL	RO, #4				
			07	12	003D1	BNEQ	55\$				
		0F	00000000G	EF	E9	003D3	BLBC	OUTOPT+8, 56\$			
				51	D5	003DA	TSTL	R1			
				39	18	003DC	BGEQ	59\$			
		50	00000000G	EF	D0	003DE	MOVL	TSF, RO			
		2E	08	A0	E9	003E5	BLBC	8(R0), 59\$			
04	00000000G	EF	04	04	ED	003E9	CMPZV	#4, #4, GCA+208, #4	0919		
				23	13	003F2	BEQL	59\$			
05	00000000G	EF	04	04	ED	003F4	CMPZV	#4, #4, GCA+208, #5			
				18	13	003FD	BEQL	59\$			
		53	00000000G	EF	D0	003FF	MOVL	OUTOPT+20, R3	0921		
				52	D4	00406	CLRL	I			
				09	11	00408	BRB	58\$			
				08	DD	0040A	PUSHL	#11	0922		
		F3	00000000G	EF	01	FB	0040C	CALLS	#1, CLH		
				52	F2	00413	AOBLSS	R3, I, 57\$			
				50	015C	CB	D0	00417	59\$:	0926	
				01	50	D1	0041C	CMPL	RO, #1		
					59	15	0041F	BLEQ	65\$		
				07	50	D1	00421	CMPL	RO, #7		
					54	18	00424	BGEQ	65\$		
				51	0158	CB	D0	00426	MOVL	LAST_PASS, R1	0927
					1F	15	0042B	BLEQ	63\$		
					52	D4	0042D	CLRL	R2		
				03	50	D1	0042F	CMPL	RO, #3		
					04	12	00432	BNEQ	60\$		
					52	D6	00434	INCL	R2		
					05	11	00436	BRB	61\$		
				04	50	D1	00438	CMPL	RO, #4		
					1E	12	0043B	BNEQ	64\$		
				05	52	E8	0043D	BLBS	R2, 62\$		
				04	50	D1	00440	CMPL	RO, #4		
					07	12	00443	BNEQ	63\$		
				0F	00000000G	EF	E9	00445	62\$:		
					51	D5	0044C	BLBC	OUTOPT+8, 64\$		
					2A	18	0044E	TSTL	R1		
					50	00000000G	EF	D0	00450	63\$:	
					1F	08	A0	E9	00457	BGEQ	65\$
04	00000000G	EF	04	04	ED	0045B	MOVL	TSF, RO			
					14	13	00464	BLBC	8(R0), 65\$		
05	00000000G	EF	04	04	ED	00466	CMPZV	#4, #4, GCA+208, #4	0930		
					09	13	0046F	BEQL	65\$		
					08	DD	00471	CMPZV	#4, #4, GCA+208, #5		
					01	FB	00473	BEQL	65\$		
04	00000000G	EF	04	04	ED	0047A	PUSHL	#11	0932		
					08	13	00483	CALLS	#1, CLH		
					04	ED	00485	CMPZV	#4, #4, GCA+208, #4	0940	
05	00000000G	EF	04	04	ED	00485	BEQL	66\$			
					09	12	0048E	CMPZV	#4, #4, GCA+208, #5		
							BNEQ	67\$			

LOHORI  
V04-000

Line output (horizontal motion)

BUILD\_LINE -- output entire text line, using mu

J 5  
16-Sep-1984 00:51:15  
14-Sep-1984 13:06:57

VAX-11 Bliss-32 V4.0-742  
[RUNOFF.SRC]LOHORI.BLI;1

Page 26  
(5)

05	015C	CB	D1	00490	66\$:	CMPL	PASS_CNTR, #5	: 0945
			24	18 00495		BGEQ	69\$	: 0947
			06	11 00497		BRB	68\$	: 0951
	0158	CB	D5	00499	67\$:	TSTL	LAST_PASS	: 0953
			1C	15 0049D		BLEQ	69\$	: 0958
	00000000G	EF	D4	0049F	68\$:	CLRL	FRA+12	: 0962
00000000G	EF	00000000G	EF	9E 004A5		MOVAB	FRA+16, FRA	
00000000G	EF	00000000G	EF	D0 004B0		MOVL	FRA, FRA+4	
00000000G	EF		00	FB 004BB	69\$:	CALLS	#0, COMPUTE_NEXT_PASS	
015C	CB		50	D0 004C2		MOVL	R0, PASS_CNTR	
			50	D4 004C7		CLRL	R0	
			04	004C9		RET		

; Routine Size: 1226 bytes, Routine Base: \$CODE\$ + 0182

LO  
VO

```

838 0963 1 %SBTTL 'compute_next_pass -- Decide which pass comes next'
839 0964 1 ROUTINE compute_next_pass =
840 0965 1
841 0966 1 !++
842 0967 1 FUNCTIONAL DESCRIPTION:
843 0968 1
844 0969 1     Decide which pass over the MRA the next pass should be.
845 0970 1
846 0971 1 FORMAL PARAMETERS:     None
847 0972 1
848 0973 1 IMPLICIT INPUTS:
849 0974 1
850 0975 1     The value of the OWN variable pass_cntr is used as the starting
851 0976 1     point for the calculation.
852 0977 1
853 0978 1     Values in the tsf, outopt, and gca structures are used (in some
854 0979 1     cases by way of macros defined at the top of this module).
855 0980 1
856 0981 1     The order of the pass ... literals defined in PASS.REQ determines
857 0982 1     the overall logic of this routine.
858 0983 1
859 0984 1 IMPLICIT OUTPUTS:     None
860 0985 1
861 0986 1 ROUTINE VALUE:
862 0987 1 COMPLETION CODES:
863 0988 1
864 0989 1     Returns the new value for pass_cntr (but does not update the OWN
865 0990 1     itself).
866 0991 1
867 0992 1 SIDE EFFECTS:         None
868 0993 1
869 0994 1 --
870 0995 1
871 0996 2 BEGIN
872 0997 2
873 0998 2 LOCAL
874 0999 2     next_one;
875 1000 2
876 1001 2     ! Start with the current value of the pass counter.
877 1002 2     next_one = .pass_cntr;
878 1003 2
879 1004 2     ! Increment based on whether there is any bolding or not.
880 1005 3     next_one = (IF .tsf_bld
881 1006 3         THEN
882 1007 4         (.next_one + 1)
883 1008 3         ELSE
884 1009 4         (.next_one + 2)
885 1010 2         );
886 1011 2
887 1012 2     ! Check for underlining in Passes 3 and 4.
888 1013 3     IF (.next_one EQL pass_underline)
889 1014 2     OR
890 1015 3     (.next_one EQL pass_bold_underline)
891 1016 2 THEN
892 1017 3     IF (NOT .tsf_und)           !No underlining to do, or
893 1018 3     OR (.outopt_und nosp)      ! non-spacing underline
894 1019 3     AND NOT .outopt_und_sep) ! (was already done)...

```

```

895 1020 THEN
896 1021     next_one = .next_one + 2; ! Skip the underlining pass.
897 1022
898 1023 ! For LN01 output, any overstriking is taken care of at the
899 1024 ! overstrike pass (only).
900 1025 IF (.next_one EQL pass_bold_overstrike)
901 1026     AND
902 1027     (laser_output)
903 1028 THEN
904 1029     next_one = pass_real_text;
905 1030
906 1031 ! Skip if no overstriking required.
907 1032 IF (.next_one GEQ pass_overstrike)
908 1033     AND
909 1034     (NOT .tsf_ovr)
910 1035 THEN
911 1036     next_one = pass_real_text;
912 1037
913 1038 RETURN .next_one;
914 1039
915 1040 END;

```

! End of compute\_next\_pass

0004 0000 COMPUTE_NEXT_PASS:										
										0964
		52	00000000G	EF	9E	00002			.WORD	Save R2
		51	00000000'	EF	D0	00009			MOVAB	GCA+208, R2
		50	00000000G	EF	D0	00010			MOVL	PASS_CNTR, NEXT_ONE
		04	08	A0	E9	00017			MOVL	TSF_RO
				51	D6	0001B			BLBC	8(R0), 1\$
				03	11	0001D			INCL	NEXT_ONE
		51		02	C0	0001F	1\$:		BRB	2\$
		03		51	D1	00022	2\$:		ADDL2	#2, NEXT_ONE
				05	13	00025			CMPL	NEXT_ONE, #3
		04		51	D1	00027			BEQL	3\$
				16	12	0002A			CMPL	NEXT_ONE, #4
	0E	08	A0	01	E1	0002C	3\$:		BNEQ	5\$
			0A 00000)00G	EF	E9	00031			BBC	#1, 8(R0), 4\$
			03 00000)00G	EF	E8	00038			BLBC	OUTOPT+4, 5\$
				02	C0	0003F	4\$:		BLBS	OUTOPT+8, 5\$
		06		51	D1	00042	5\$:		ADDL2	#2, NEXT_ONE
				11	12	00045			CMPL	NEXT_ONE, #6
04	62	04		04	ED	00047			BNEQ	7\$
				07	13	0004C			CMPZV	#4, #4, GCA+208, #4
05	62	04		04	ED	0004E			BEQL	6\$
				03	12	00053			CMPZV	#4, #4, GCA+208, #5
				07	D0	00055	6\$:		BNEQ	7\$
				51	D1	00058	7\$:		MOVL	#7, NEXT_ONE
				08	19	0005B			CMPL	NEXT_ONE, #5
				02	E0	0005D			BLSS	8\$
	03	08	A0	07	D0	00062			BBS	#2, 8(R0), 8\$
				51	D0	00065	8\$:		MOVL	#7, NEXT_ONE
				50	D0	00068			MOVL	NEXT_ONE, R0
				04	00068				RET	1040

LOHORI  
V04-000

Line output (horizontal motion)  
compute\_next\_pass -- Decide which pass comes ne

M 5  
16-Sep-1984 00:51:15  
14-Sep-1984 13:06:57

VAX-11 Bliss-32 V4.0-742  
[RUNOFF.SRC]LOHORI.BLI;1

Page 29  
(6)

LO  
VO

: Routine Size: 105 bytes, Routine Base: \$CODE\$ + 064C

: 916 1041 1  
: 917 1042 1 END ! End of module  
: 918 1043 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	352	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1717	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	20	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	0	0	252	00:00.1
_\$255\$DUA28:[RUNOFF.SRC]DSRLIB.L32;1	1248	67	5	86	00:00.2

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:LOHORI/OBJ=OBJ\$:LOHORI MSRC\$:LOHORI/UPDATE=(ENH\$:LOHORI)

: Size: 1717 code + 372 data bytes  
: Run Time: 00:27.2  
: Elapsed Time: 01:00.6  
: Lines/CPU Min: 2304  
: Lexemes/CPU-Min: 17293  
: Memory Used: 259 pages  
: Compilation Complete

0343 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY


LOEMPH LIS

LOHORI LIS

LOVERT LIS

LPI LIS

LSTOPS LIS

MAKNDX LIS

NATE LIS

NOXDAT LIS

NOXEMT LIS