

.REM 8

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

IDENTIFICATION

PRODUCT CODE: AC-8954E-MC
PRODUCT NAME: CZM9AE0 ROOTSTRAP/TERMINATOR (M9301, M9400)
PROGRAM DATE: APRIL, 1979
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1976, 1979 BY DIGITAL EQUIPMENT CORPORATION

* SUMMARY OF OPERATING INSTRUCTIONS *

THE FOLLOWING PROCEDURE CAN BE USED TO RUN THIS DIAGNOSTIC
IN A DEVICE VERIFICATION MODE. IF THE PROGRAM DOES NOT
RUN SUCCESSFULLY CONSULT THE FOLLOWING DOCUMENT FOR ASSISTANCE.

OPERATING PROCEDURE:

1. LOAD THE PROGRAM USING NORMAL PROCEDURES.
2. LOAD ADDRESS 200
3. SET SWITCH REGISTER TO SELECT THE PROPER
VERSION OF THE ROM UNDER TEST.
(SEE INSTRUCTIONS IF A SOFTWARE SWITCH REGISTER
IS TO BE USED.)
4. PRESS START
5. THE PROGRAM SHOULD TAKE ABOUT 1 SEC TO
COMPLETE THE TEST AND PRINT: "END OF TEST".
6. IF THE PROGRAM DOES NOT RUN AS DESCRIBED
ABOVE, CONSULT THE FULL OPERATING INSTRUCTIONS
WHICH FOLLOW.

* * CAUTION * *

BECAUSE THE CONTENTS READ FROM LOCATION 773024
OF THE M9301 OPTION IS CONFIGURATION DEPENDANT(SWITCH
REGISTER DEPENDANT), THIS LOCATION IS NOT INCLUDED IN
THE DATA CHECK.
THIS LOCATION CAN BE VERIFIED BY EXAMINING IT OR BY USING
THE ALTERNATE STARTING ADDRESS(SEE SECTION 2.1.3) TO
PRINT OUT THIS LOCATION.

50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90

91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144

1.0 GENERAL PROGRAM INFORMATION

1.1 PROGRAM PURPOSE

THIS DIAGNOSTIC PROGRAM IS INTENDED TO VERIFY THE ROM CONTENTS OF THE ROM BOOTSTRAP MODULES. THE PROGRAM COMPUTES AND CHECKS A CYCLIC REDUNDANCY CHARACTER AND A LONGITUDINAL PARITY CHARACTER FOR THE CONTENTS OF THE ROM STORAGE AVAILABLE IN AN M9301 OR M9400 MODULE.

A SEPARATE ROUTINE INCLUDED ALLOWS THE USER TO TYPE THE CONTENTS OF THE ROM STORAGE ON THE TELETYPE AS AN AID TO DEBUGGING.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE

PDP/11 PROCESSOR
TELETYPE OR EQUIVALENT
4K OF MEMORY
M9301 OR M9400 MODULE

1.2.2 SOFTWARE

THIS PROGRAM IS WRITTEN TO BE RUN AS A STAND-ALONE PROGRAM. HOWEVER, THE PROGRAM IS DESIGNED TO RUN UNDER AUTOMATED PRODUCT TEST SYSTEM (APT) IN ALL THREE MODES.

THE PROGRAM CAN ALSO BE RUN UNDER THE ACT 11 MONITOR.

1.3 RELATED DOCUMENTS AND STANDARDS

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS PROGRAMMING PRACTICES
DOCUMENT NO. 175-003-009-00

APT INTERFACE SPECIFICATION, REV. 13

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE, HOWEVER THE CPU IS ASSUMED TO BE FUNCTIONING

1.5 FAILURE ASSUMPTIONS

THE PROCESSOR IS ASSUMED TO BE FUNCTIONING PROPERLY.

145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOADING

USE NORMAL PROCEDURES FOR LOADING DIAGNOSTIC PROGRAMS.

2.1.2 NORMAL START

1. LOAD SOFTWARE SWITCH REGISTER (IF USED) TO SELECT THE ROM VERSION UNDER TEST. (SEE 2.3)

2. LOAD ADDRESS 200

3. SET HARDWARE SWITCH REGISTER (IF AVAILABLE) TO SELECT THE ROM VERSION UNDER TEST (SEE 2.3).

4. START

2.1.3 OPTIONAL START

THE OPTIONAL STARTING ADDRESS IS USED TO TYPE OUT THE CONTENTS OF THE ROM FOR USE IN VISUAL VERIFICATION OR AS A DEBUGGING TOOL.

USE THE SAME PROCEDURE AS A NORMAL START EXCEPT USE ADDRESS 210 IN STEP 2.

2.2 SPECIAL ENVIRONMENTS

THIS PROGRAM IS WRITTEN TO COMPLY WITH ALL THE REQUIREMENTS OF THE APT INTERFACE SPECIFICATION. IT WILL RUN UNDER APT IN EITHER QUICK VERIFY, PROGRAM OR RUN-TIME MODES.

THIS PROGRAM IS WRITTEN TO COMPLY WITH THE ACT11/XXDP INTERFACE REQUIREMENTS.

WHEN RUNNING IN ACT11 QUICK VERIFY OR XXDP CHAIN MODE (LOC. 42 NOT 0), OR APT QUICK VERIFY AND PROGRAM MODE THE PROGRAM ATTEMPTS TO RUN WITHOUT OPERATOR INTERVENTION OR SWITCH REGISTER SELECTION. THE COMPUTED CRC IS COMPARED AGAINST THE CRC FOR ALL KNOWN VERSIONS OF THE ROM BOOTSTRAP. WHEN A MATCH IS FOUND THE VERSION OF THE MODULE IS TYPED FOR VISUAL VERIFICATION.

198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253

2.3 PROGRAM OPTIONS

THE PROGRAM AUTOMATICALLY CHECKS FOR THE PRESENCE OF A HARDWARE SWITCH REGISTER. IF NO RESPONSE IS FOUND WHEN ADDRESSING THE HARDWARE SWR (177570), THE ADDRESS OF THE SOFTWARE SWR (176) IS SUBSTITUTED.

FOR PROCESSORS WITH NO HARDWARE SWITCH REGISTER, THE OPERATOR SHOULD SET THE DESIRED SWITCH VALUE IN LOCATION 176

WARNING... IN ORDER TO ALLOW TESTING OF M7942-YB BOARDS ON THE VT71, IF LOCATION 176 IS SET TO 6, THE SOFTWARE SWITCH REGISTER WILL BE USED REGARDLESS OF HARDWARE SWITCH REGISTER AVAILABILITY.

2.3.1 SWITCH SELECTION

THE SWITCH REGISTER (HARDWARE OR SOFTWARE) IS USED TO SELECT THE VERSION OF THE ROM BOOTSTRAP TO BE TESTED ACCORDING TO THE FOLLOWING TABLE. NOTE: THESE SETTINGS ARE OCTAL NUMBERS. THEY ARE NOT PARTICULAR SWITCHES SET TO A ONE. FOR EXAMPLE, TO SELECT THE M9301-YH VERSION, SET SWITCHES #3 AND #1 IN THE SWITCH REGISTER. THIS CORRESPONDS TO AN OCTAL 12.

SWR	MODULE VERSION
1	M9301-YA
2	M9301-YB
3	M9301-YC
4	M9400-YA (OR YC)
5	M9301-YF
6	M7942-YR
7	M9301-YD
10	M9400-YH (OR YK)
11	M9311 REVISION *
12	M9301-YH
13	M9301-YE
14	M9301-YJ
15	M9400-YN
16	M9311 REVISION A ***

* --- SEE NOTE BELOW ---

NOTE: FOR M9311 ONLY.....
IF ROM IN LOCATION E-5 IS #23-524A9, THEN YOU HAVE REVISION *. (I.E. USE SWR 11 SETTING)
IF ROM IN LOCATION E-5 IS #23-688A9, THEN YOU HAVE REVISION A. (I.E. USE SWR 16 SETTING)

IF THE CRC AND LPC FOR NEW VERSIONS ARE KNOWN PUT NOT IN THE ABOVE TABLE, SET THE SWITCH REGISTER TO ZERO AND ANSWER THE TELETYPE DIALOG.

254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309

TO DETERMINE THE CRC AND LPC FOR A NEW VERSION. START THE DIAGNOSTIC AT 200 WITH SWP=0, ANSHEK 0 TO THE REQUESTS FOR THE LPC AND CRC. THE RESULTING MESSAGES WILL INDICATE THE CORRECT FUTURE RESPONSES FOR CRC AND LPC PROVIDED THE TEST IS RUN ON A KNOWN-GOOD MODULE.

2.3.2 TFLTYPE DIALOG

SEVERAL QUESTIONS ARE ASKED OF THE OPERATOR IN ORDER TO OBTAIN SUFFICIENT INFORMATION FOR TESTING A ROM MODULE NOT PREVIOUSLY SUPPORTED IN THE DIAGNOSTIC. THE DIALOG IS INITIATED IF THE PROGRAM IS STARTED WITH THE SWR = 0. ALL RESPONSES ARE IN OCTAL AND TERMINATED BY A CARRIAGE RETURN.

ALL RESPONSES ARE CHECKED FOR VALID OCTAL NUMBERS. IF AN ILLEGAL CHARACTER IS TYPED, THE PROGRAM WILL TYPE A "?", CARRIAGE RETURN-LINE FEED AND AWAIT THE PROPER INPUT.

IF A MISTAKE IS NOTICED BEFORE THE CARRIAGE RETURN IS USED TO TERMINATE THE INPUT, A RUBOUT CAN BE USED TO DELETE MISTYPED INPUT.

1. TYPE CRC VALUE:
THIS REQUESTS THE VALUE OF THE CYCLIC REDUNDANCY CHECK PREVIOUSLY CALCULATED FOR THIS VERSION OF THE ROM MODULE. IT IS THE VALUE AGAINST WHICH THE UNIT UNDER TEST'S CRC WILL BE COMPARED.

2. TYPE LPC VALUE:
THIS REQUESTS THE VAULE OF THE LONGITUDINAL PARITY CHECK PREVIOUSLY CALCULATED FOR THIS VERSION OF THE ROM MODULE. IT IS THE VALUE AGAINST WHICH THE UNIT UNDER TEST'S LPC WILL BE COMPARED.

3. TYPE STARTING ADDR. OF 1ST ROM ADDR. SPACE:
THIS QUESTION REFERS TO THE FACT THAT THE ROM SPACE IN AROM BOOTSTRAP MODULE IS DIVIDED INTO 2 DISTINCT ADDRESS SPACES. TYPE THE STARTING ADDRESS OF THE 1ST RANGE OF ADDRESSES. THE STANDARD M9301 & M9400 BEGIN AT 173000.

4. TYPE LENGTH (BYTES) OF 1ST ROM ADDR. SPACE:
THIS REQUESTS THE LENGTH OF THE 1ST GROUP OF ROM ADDRESSES IN BYTES. THE STANDARD M9301 & M9400 HAVE AN INITIAL ADDRESS SPACE OF 1000 BYTES. IF THIS SECTION OF ADDRESSES IS NOT USED BY THIS VERSION, ANSWER 0 TO THIS QUESTION.

5. TYPE STARTING ADDR. OF 2ND ROM ADDR. SPACE:
THIS REFERS TO THE FIRST ADDRESS IN THE SECOND DISTINCT GROUP OF ROM ADDRESSES. THE RESPONSE FOR A STANDARD M9301 & M9400 WOULD BE 165000.

6. TYPE LENGTH (BYTES) OF 2ND ROM ADDR. SPACE:
THIS REQUESTS THE LENGTH OF THE 2ND GROUP OF ROM ADDRESSES IN BYTES. THE STANDARD M9301 & M9400 HAVE A SECOND ADDRESS SPACE

310
311
312
313
314
315
316
317
318
319

OF 1000 BYTES. IF THIS SECTION OF ADDRESSES IS NOT USED
BY THIS VERSION, ANSWER 0 TO THIS QUESTION.

2.4 EXECUTION TIMES

THE DIAGNOSTIC COMPLETES 1 PASS IN LESS THAN 1 SEC.
ONCE THE INPUT DIALOG HAS BEEN COMPLETED.
THE PROGRAM WILL HALT UPON COMPLETION; HOWEVER, IF RUNNING
UNDER APT THE PROGRAM WILL CYCLE CONTINUOUSLY.

320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375

3.0

ERROR INFORMATION

WHEN THE DIAGNOSTIC DETECTS A DISCREPANCY BETWEEN THE EXPECTED AND COMPUTED CRC OR LPC BOTH ARE PRINTED ON THE TELETYPE.

ANY DISCREPANCY IN THE LPC CAN ASSIST IN ISOLATING THE PROBLEM TO THE ROM IC.

UNDER APT THE ERROR IS INDICATED BY DEPOSITING ERROR INFORMATION IN THE APT MAILBOX BEFORE HALTING.

4.0

PROGRESS REPORTS

AT THE END OF EACH PASS THE PROGRAM INCREMENTS TO THE LOCATION \$PASS WHICH IS IN THE APT MAILBOX. THIS LOCATION WILL ALWAYS CONTAIN THE NUMBER OF PASSES COMPLETED. \$PASS IS RESET WITH EVERY RESTART.

ADDITIONALLY, THE MESSAGE "END OF TEST" IS PRINTED ON THE CONSOLE TELETYPE AFTER EACH PASS. NORMALLY ONLY ONE PASS NEEDS TO RUN TO VERIFY THE MODULE.

5.0

TROUBLE SHOOTING

THE ALGORITHM FOR COMPUTING THE CRC IS THE SAME AS THAT USED ON 9-TRACK MAGNETIC TAPE WITH ODD PARITY. THE CRC IS CALCULATED ON A BYTE-BY-BYTE BASIS. WHILE THE ALGORITHM IS SUCCESSFUL IN DETECTING MULTIPLE ERRORS, ITS USE AS A DEBUGGING AID IS LIMITED.

THE LPC IS CALCULATED BY ASSEMBLING THE XOR OF EVERY WORD IN THE ROM. WHILE ONLY USEFUL IN CATCHING AN ODD NUMBER OF ERRORS IN EACH BIT POSITION, IT IS VERY USEFUL IN ISOLATING THE PROBLEM TO A CHIP. BY LOCATING WHICH BIT POSITIONS ARE IN DISCREPANCY, THE CORRESPONDING ROM CHIPS CAN BE ISOLATED.

IF NO OTHER CLUES CAN BE OBTAINED, START THE PROGRAM AT 210 AND COMPARE THE PRINTOUT OF THE CODE WITH THE LISTING FOR THE VERSION BEING TESTED.

* * CAUTION * *

BECAUSE THE CONTENTS READ FROM LOCATION 773024 OF THE M9301 OPTION IS CONFIGURATION DEPENDANT(SWITCH REGISTER DEPENDANT), THIS LOCATION IS NOT INCLUDED IN THE DATA CHECK.

THIS LOCATION CAN BE VERIFIED BY EXAMINING IT OR BY USING THE ALTERNATE STARTING ADDRESS(SEE SECTION 2.1.3) TO PRINT OUT THIS LOCATION.

376
377
378
379
380
381
382
383
384

6,0 LISTING

1

385
386 .ENABLE ABS
387 .LIST ME
388 .MLIST MC,MD,CMD
389 000000 \$SWR=0
390 .SRTTL BASIC DEFINITIONS
391
392 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
393 STACK= 1100
394 .EQUIV EMT,ERROR ;*BASIC DEFINITION OF ERROR CALL
395 .EQUIV IOT,SCOPE ;*BASIC DEFINITION OF SCOPE CALL
396
397 ;*MISCELLANEOUS DEFINITIONS
398 000011 HT= 11 ;*CODE FOR HORIZONTAL TAB
399 000012 LF= 12 ;*CODE FOR LINE FEED
400 000015 CR= 15 ;*CODE FOR CARRIAGE RETURN
401 000200 CRLF= 200 ;*CODE FOR CARRIAGE RETURN+LINE FEED
402 177776 PS= 177776 ;*PROCESSOR STATUS WORD
403 .EQUIV PS,PSW
404 177774 STKLM= 177774 ;*STACK LIMIT REGISTER
405 177772 PIRQ= 177772 ;*PROGRAM INTERRUPT REQUEST REGISTER
406 177570 DSWR= 177570 ;*HARDWARE SWITCH REGISTER
407 177570 DDISP= 177570 ;*HARDWARE DISPLAY REGISTER
408
409 ;*GENERAL PURPOSE REGISTER DEFINITIONS
410 000000 R0= %0 ;*GENERAL REGISTER
411 000001 R1= %1 ;*GENERAL REGISTER
412 000002 R2= %2 ;*GENERAL REGISTER
413 000003 R3= %3 ;*GENERAL REGISTER
414 000004 R4= %4 ;*GENERAL REGISTER
415 000005 R5= %5 ;*GENERAL REGISTER
416 000006 R6= %6 ;*GENERAL REGISTER
417 000007 R7= %7 ;*GENERAL REGISTER
418 000006 SP= %6 ;*STACK POINTER
419 000007 PC= %7 ;*PROGRAM COUNTER
420
421 ;*PRIORITY LEVEL DEFINITIONS
422 000000 PR0= 0 ;*PRIORITY LEVEL 0
423 000040 PR1= 40 ;*PRIORITY LEVEL 1
424 000100 PR2= 100 ;*PRIORITY LEVEL 2
425 000140 PR3= 140 ;*PRIORITY LEVEL 3
426 000200 PR4= 200 ;*PRIORITY LEVEL 4
427 000240 PR5= 240 ;*PRIORITY LEVEL 5
428 000300 PR6= 300 ;*PRIORITY LEVEL 6
429 000340 PR7= 340 ;*PRIORITY LEVEL 7
430
431 ;*SWITCH REGISTER SWITCH DEFINITIONS
432 100000 SW15= 100000
433 040000 SW14= 400000
434 020000 SW13= 200000
435 010000 SW12= 100000
436 004000 SW11= 400000
437 002000 SW10= 200000
438 001000 SW09= 100000
439 000400 SW08= 400000
440 000200 SW07= 200000

```
441 000100 SW06= 100
442 000040 SW05= 40
443 000020 SW04= 20
444 000010 SW03= 10
445 000004 SW02= 4
446 000002 SW01= 2
447 000001 SW00= 1
448 .EQUIV SW09,SW9
449 .EQUIV SW08,SW8
450 .EQUIV SW07,SW7
451 .EQUIV SW06,SW6
452 .EQUIV SW05,SW5
453 .EQUIV SW04,SW4
454 .EQUIV SW03,SW3
455 .EQUIV SW02,SW2
456 .EQUIV SW01,SW1
457 .EQUIV SW00,SW0
458
459 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
460 BIT15= 100000
461 BIT14= 40000
462 BIT13= 20000
463 BIT12= 10000
464 BIT11= 4000
465 BIT10= 2000
466 BIT09= 1000
467 BIT08= 400
468 BIT07= 200
469 BIT06= 100
470 BIT05= 40
471 BIT04= 20
472 BIT03= 10
473 BIT02= 4
474 BIT01= 2
475 BIT00= 1
476 .EQUIV BIT09,BIT9
477 .EQUIV BIT08,BIT8
478 .EQUIV BIT07,BIT7
479 .EQUIV BIT06,BIT6
480 .EQUIV BIT05,BIT5
481 .EQUIV BIT04,BIT4
482 .EQUIV BIT03,BIT3
483 .EQUIV BIT02,BIT2
484 .EQUIV BIT01,BIT1
485 .EQUIV BIT00,BIT0
486
487 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
488 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
489 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
490 TRIVEC=14 ;;"T" BIT
491 TRIVEC= 14 ;;TRAP TRAP
492 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
493 IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOI) **SCOPE**
494 PWRVEC= 24 ;;POWER FAIL
495 EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
496 TRAPVEC=34 ;;"TRAP" TRAP
```

```
497 000060 TKVEC= 60 ;;TTY KEYBOARD VECTOR
498 000064 TPVEC= 64 ;;TTY PRINTER VECTOR
499 000240 PIQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
500 000000 .=0
501 .SBTTL TRAP CATCHER
502 .=0
503 000000 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "+2,HALT"
504 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
505 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
506
507 000174 .=174
508 000174 000000 DISPREG: ,WORD 0 ;;SOFTWARE DISPLAY REGISTER
509 000176 000000 SWREG: ,WORD 0 ;;SOFTWARE SWITCH REGISTER
510 000200 .=200
511 000200 000624 CLR TYP0UT
512 000204 000167 000666 JMP START
513 000210 012767 000001 000612 MOV #1,TYP0UT
514 000216 000167 000654 JMP START
515
516 177776 PS=177776
517 000034 TRAPVEC=34
518
519 001000 .=1000
520 001000 177570 SWR: 177570
521 001002 177570 DISPLAY: 177570
522 001004 173000 ROMSA1: 173000
523 001006 001000 DATLN1: 512
524 001010 165000 ROMSA2: 165000
525 001012 001000 DATLN2: 512
526 001014 000000 XORS: 0
527 001016 000000 EXCRC: 0
528 001020 000000 EXLPC: 0
529 001022 000000 ACTCRC: 0
530 001024 000000 ACTLPC: 0
531 001026 000000 PARCNT: 0
532 001030 000000 TYP0UT: 0
533
534 .SBTTL ACT11 HOOKS
535
536 ;*****
537 ;HOOKS REQUIRED BY ACT11
538 001032 000000 .=SVPC=, ;SAVE PC
539 000046 002060 .=46 .ENDAD ;;1)SET LOC,46 TO ADDRESS OF .ENDAD IN .SEOP
540 000052 000000 .=52 .WORD 0 ;;2)SET LOC,52 TO ZERO
541 001032 .=SVPC ;; RESTORE PC
542 .SBTTL APT PARAMETER BLOCK
543
544 ;*****
545 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
546 ;*****
547 .=24 ;;SAVE CURRENT LOCATION
548 .=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
549 000024 000200 200 ;;FOR APT START UP
550 000044 .=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
```

```
553 000044 001032          $APTHDR ;;POINT TO APT HEADER BLOCK
554          001032          ,*,SX ;;RESET LOCATION COUNTER
555
556 ;;*****
557 ;;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
558 ;;INTERFACE SPEC.
559
560 001032          $APTHDR
561 001032 000000          $HIBTSI ,WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
562 001034 001016          $HADDRI ,WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
563 001036 000002          $STMTI ,WORD 2 ;;RUN TIME OF LONGEST TEST
564 001040 000002          $PASTMI ,WORD 2 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
565 001042 000000          $UNITMI ,WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
566 001044 000014          $UNITMI ,WORD $TEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(*WORDS)
567
568 ,SRTTL APT MAILBOX-ETABLE
569
570 ;;*****
571 ,EVEN
572 $MAILI
573 $MSGTYI ,WORD $MSGTY ;;MESSAGE TYPE CODE
574 $FATALI ,WORD $FATAL ;;FATAL ERROR NUMBER
575 $TESTNI ,WORD $TESTN ;;TEST NUMBER
576 $PASSI ,WORD $PASS ;;PASS COUNT
577 $DEVCTI ,WORD $DEVCT ;;DEVICE COUNT
578 $UNITI ,WORD $UNIT ;;I/O UNIT NUMBER
579 $MSGADI ,WORD $MSGAD ;;MESSAGE ADDRESS
580 $MSGLGI ,WORD $MSGLG ;;MESSAGE LENGTH
581 $ETABLEI
582 $ENVI ,BYTE $ENV ;;ENVIRONMENT BYTE
583 $ENVM ,BYTE $ENVM ;;ENVIRONMENT MODE BITS
584 $SWREGI ,WORD $SWREG ;;APT SWITCH REGISTER
585 $USWRI ,WORD $USWR ;;USER SWITCHES
586 $CPUUPI ,WORD $CPUUP ;;CPU TYPE,OPTIONS
587 ;;
588 ;; BITS 15-11=CPU TYPE
589 ;;
590 ;; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
591 ;; 11/70=06,000=07,0=10
592 ;;
593 ;; BIT 10=REAL TIME CLOCK
594 ;;
595 ;; BIT 9=FLOATING POINT PROCESSOR
596 ;;
597 ;; BIT 8=MEMORY MANAGEMENT
598
599 $TENDI
600 ,MEXIT
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
```

```
609 001170 012767 177570 177604 MOV $DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
610 001176 022777 177777 177574 CMP #1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
611 001204 001012 BNE 665 ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
612
613 001206 000403 ER 655 ;;AND THE HARDWARE SWR IS NOT = -1
614 001210 012716 001216 648: MOV #656,($SP) ;;BRANCH IF NO TIMEOUT
615 001214 000002 RTI ;;SET UP FOR TRAP RETURN
616 001216 012767 000176 177554 658: MOV $SWREG,SWR ;;POINT TO SOFTWARE SWR
617 001224 012767 000174 177550 MOV $DISPREG,DISPLAY
618 001232 012637 000104 668: MOV ($SP)+,$ERRVEC ;;RESTORE ERROR VECTOR
619
620 001236 005067 177612 CLR $PASS ;;CLEAR PASS COUNT
621 001242 132767 000200 177617 BITR $APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
622 001250 001403 BEQ 678 ;;YES,USE NON-API SWITCH
623 001252 012767 001070 177520 MOV $SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
624 001260 026727 176556 002060 678: CMP 42,$SENDAD ;;ACT AUTO MODE?
625 001266 001402 BEQ RESTRT ;;IF SO: BR
626 001270 104401 TYPE
627 001272 005276 TITL
628
629
630 001274 123727 000176 000006 RESTRT: CMPB #176,#6 ;;IS SOFTWARE SWITCH REGISTER =6?
631 001302 001003 BNE 18 ;;IF NOT, TEST NORMALLY
632 001304 012767 000176 177466 MOV $SWREG,SWR ;;IF SO USE THE SOFTWARE SWITCH REG
633 001312 017700 177462 18: MOV $SWR,#0 ;;GET SWR
634 001316 001424 RFO GETIN ;;IF ZEP0: GET INPUT
635 001320 006300 ST2: ASL R0
636 001322 016067 004714 177470 MOV TXLPC(R0),EXLPC ;;FETCH EXPECT, LPC
637 001330 016067 004644 177460 MOV TXCRC(R0),EXCRC ;;FETCH EXPECTED CRC
638 001336 016067 004766 177442 MOV TDLN1(R0),DAILN1 ;;FETCH 1ST LENGTH
639 001344 016067 005036 177432 MOV TRMSA1(R0),ROMSA1 ;;FETCH 1ST STARTING ADDR.
640 001352 016067 005106 177432 MOV TDLN2(R0),DAILN2 ;;FETCH 2ND LENGTH
641 001360 016067 005156 177422 MOV TRMSA2(R0),ROMSA2 ;;FETCH 2ND STARTING ADDR
642 001366 000450 BF ;;GO COMPUTE
643 001370 005767 176446 GETIN: TST 42 ;;UNDER ACT AUT ACCEPT?
644 001374 001045 RNE CHECK ;;IF SO: BR USE DEFAULT PARAMETERS
645 001376 122767 000001 177462 CNPR #1,$ENV ;;UNDER APT?
646 001404 001441 BEQ CHECK ;;IF SO: BR
647 001406 005767 177416 TST TYP0UT ;;FROM TYPE OPTION
648 001412 001012 RNE GET2 ;;IF SO: BR
649
650 TYPE
651 GETCRC
652 RDOCT
653 MOV ($SP)+,EXCRC ;;STORE EXPECT, CRC
654 TYPE LPC INPUT REQUEST
655 GETLPC
656 RDOCT
657 MOV ($SP)+,EXLPC ;;STORE EXPECTED LPC
658 TYPE SA1
659 RDOCT
660 MOV ($SP)+,ROMSA1 ;;REQUEST 1ST ADDRESS SPACE
661 TYPE SA
662 MOV ($SP)+,DAILN1 ;;REQUEST LENGTH OF 1ST ADDR, SPACE
663 SIZE1
664 RDOCT
665 MOV ($SP)+,DAILN1 ;;INPUT LENGTH
```

```
665 001464 104401 TYPE ;REQUEST START ADDR, FOR 2ND SPACE
666 001466 005575 SA2
667 001473 104407 RDOCT ;INPUT SA
668 001472 012667 177312 MOV (SP)+,ROMSA2
669 001476 104401 TYPE ;REQUEST LENGTH OF 2ND SPACE
670 001400 005735 SIZE2
671 001502 104407 RDOCT ;INPUT LENGTH
672 001504 012667 177302 MOV (SP)+,DATLN2
673 001510 005767 177314 CHECK1 TST TYP0UT ;IS TYP0UT REQUESTED?
674 001514 001402 BEQ IS ;BRANCH IF NOT
675 001516 000167 000660 JMP TYPROM ;GO TYP OUT POM
676 001522 005067 177274 10: CLR ACTCRC ;CLEAR STORAGE FOR ACTUAL CRC
677 001526 005067 177272 CLR ACTLPC ;CLEAR STORAGE FOR ACTUAL LPC
678 001532 016700 177250 DATLN1,R0 ;SET LENGTH OF 1ST ROM SPACE
679 001536 001413 BEQ CH0 ;IF NO VERSION SELECTED: BR
680 001540 016701 177240 MOV ROMSA1,R1 ;POINT TO START OF 1ST ROM SPACE
681 001544 004767 000324 JSR PC,CRC ;COMPUTE FIRST HALF OF CRC
682 001550 016701 177230 MOV ROMSA1,R1 ;POINT TO START OF 1ST ROM ADDR.
683 001554 016700 177226 MOV DATLN1,R0 ;SET LENGTH OF 1ST ROM ADDR.
684 001560 006200 ASR R0 ;CONVERT TO WORDS
685 001562 004767 000556 JSR PC,LPC ;COMPUTE FIRST HALF OF CRC
686 001566 016701 177216 CH0: MOV ROMSA2,R1 ;POINT TO 2ND ROM ADDR.
687 001572 016700 177214 MOV DATLN2,R0 ;SET LENGTH OF 2ND ROM ADDR.
688 001576 001422 BEQ CH1 ;BR IF THIS SPACE NOT USED
689 001600 004767 000270 JSR PC,CRC ;COMPUTE REMAINDER OF CRC
690 001604 016701 177200 MOV ROMSA2,R1 ;POINT TO START OF 2ND ROM ADDR.
691 001610 016700 177176 MOV DATLN2,R0 ;SET LENGTH OF 2ND ROM ADDR.
692 001614 006200 ASR R0 ;CONVERT TO WORDS
693 001616 004767 000522 JSR PC,LPC ;COMPUTE REMAINDER OF LPC
694 001622 122767 000001 177236 CMPB #1,6ENV ;UNDER APT?
695 001630 001403 BEQ IS ;IF SO: BR
696 001632 005767 176204 TST 42 ;UNDER ACT AUTO ACCEPT?
697 001636 001402 BEQ CH1 ;IF NOT: BR
698 001640 000167 000656 10: JMP AUTACT
699 001644 026767 177146 10: CMP EXCRC,ACTCRC ;COMPUTED = EXPECTED ?
700 001652 001431 BEQ CK1 ;IF SO: BR
701
702 001654 104401 TYPE ;TYPE CRC ERROR MESSG.
703 001656 005361 EXCRMG
704 001660 016746 177132 MOV EXCRC,-(SP) ;PUT EXPECTED CRC ON STACK
705 001664 104402 TYPOC ;TYPE EXPECTED CRC
706 001666 104401 TYPE ;TYPE ACTUAL CRC MESSG
707 001670 005430 ACCRMG
708 001672 016746 177124 MOV ACTCRC,-(SP) ;PUT ACTUAL CRC ON STACK
709 001676 104402 TYPOC ;TYPE ACTUAL CRC
710 001700 026727 176136 002060 CMP 42,#SENDAD ;UNDER ACT AUTO MODE?
711 001706 001404 BEQ IS ;IF SO: BR
712 001710 122767 000001 177150 10: CMPB #1,6ENV ;UNDER APT?
713 001716 001007 RNE CK1 ;IF NOT: BR
714 001720 012767 000002 177122 10: MOV #2,$FATAL ;MOVE TO MAILBOX ERROR NO. **** 2 ****
715 001726 012767 000001 177112 10: MOV #1,$MSGTYP ;SET MAILBOX FOR FATAL ERROR
716 001734 000000 HALT ;CRC ERROR
717
718 001736 026767 177062 177054 CK1: CMP ACTLPC,EXLPC ;COMPARE EXPT. LPC=ACTUAL LPC
719 001744 001431 BEQ CK2 ;IF SO: BR
720 001746 104401 TYPE ;TYPE LPC ERROR MESSG,
```

```
721 001750 005404 EXLPMG
722 001752 016746 177042 MOV EXLPC,-(SP) ;PUT EXPECTED LPC ON STACK
723 001756 104402 TYPOC ;TYPE EXPECTED LPC
724 001760 104401 TYPE ;TYPE ACTUAL LPC MESSG.
725 001762 005453 ACLPMG
726 001764 016746 177034 MOV ACTLPC,-(SP) ;PUT ACTUAL LPC ON STACK
727 001770 104402 TYPOC ;TYPE ACTUAL LPC
728 001772 026727 176044 002060 CMP 42,#SENDAD ;UNDER ACT AUTO MODE?
729 002000 001404 BEQ IS ;IF SO: BR
730 002002 122767 000001 177056 10: CMPB #1,6ENV ;UNDER APT?
731 002010 001007 RNE CK2 ;IF NOT: BR
732 002012 012767 000003 177030 10: MOV #3,$FATAL ;MOVE TO MAILBOX ERROR NO. **** 3 ****
733 002020 012767 000001 177020 10: MOV #1,$MSGTYP ;SET MAILBOX FOR FATAL ERROR
734 002026 000000 HALT ;LPC ERROR
735
736 002030 026727 176006 002060 CK2: CMP 42,#SENDAD ;ACT AUTO ACCEPT?
737 002036 001402 BEQ IS ;IF SO: BR
738 002040 104401 TYPE ;TYPE END OF TEST
739 002042 005476 EOTST
740 002044 005267 177004 10: INC $PASS ;BUMP PASS COUNT
741 002050 013700 000042 MOV #42,R0 ;CHECK APT
742 002054 001405 BEQ GOAGIN ;KEEP GOING
743 002056 000005 RESET
744 002060 004710 0:ENDAD: JSR PC,(R0) ;ACT HOOKS
745 002062 000240 NOP
746 002064 000240 NOP
747 002066 000240 NOP
748 002070 000167 177200 GOAGIN: JMP RESTR ;DO AGAIN
749
750 002074 016767 176722 176712 CRC: MOV ACTCRC,XORS
751 002102 111104 CL0: MOVB (R1),R4
752 002104 022701 173024 CMP #173024,P1 ;LOCATION EFFECTED BY SWITCHES
753 002110 001004 RNE CL3 ;IF NOT: BR
754 002112 005300 DEC R0 ;FIX COUNTERS
755 002114 005300 DEC R0
756 002116 005721 TST (R1)+ ;FIX POINTER
757 002120 000770 RR CL0 ;CONTINUE
758 002122 004767 000114 CL3: JSR PC,PARITY ;GO GET PARITY
759 002126 004767 000166 JSR PC,XOR ;XOR CHAR
760 002132 000241 CLC
761 002134 006004 ROR R4 ;ROTATE 1 POS, RIGHT
762 002136 103014 BCC CL2 ;IF NO CARRY: BR
763 002140 052704 000400 BIS #400,R4 ;SET BIT NINE
764 002144 000241 CLC
765 002146 010405 CL1: MOV R4,R5 ;SAVE CHAR
766 002150 042705 177703 BIC #177703,P5
767 002154 005105 COM R5
768 002156 042705 177703 BIC #177703,R5
769 002162 042704 000074 BIC #74,R4
770 002166 050504 BIS R5,R4
771 002170 010467 176620 CL2: MOV R4,XORS
772 002174 005300 DEC R0
773 002176 001402 BEQ CLLAST ;IF LAST CHAR: BR
774 002200 000167 177676 JMP CL0 ;GET NEXT CHAR.
775 002204 016704 176604 CLLAST: MOV XORS,R4
776 002210 005167 176600 COM XORS
```

```

777 002214 042767 177050 176572 BIC #177050,XORS
778 002222 042704 177727 PIC #177727,R4 ;COMPLEMENT ALL BUT BITS 3 & 5
779 002226 050467 176562 BIS R4,XORS
780 002232 016767 176556 176562 MOV XORS,ACTCRC
781 002240 000207 RTS PC
782 002242 005067 176560 PARITY: CLR PARCNT ;CLEAR BIT COUNTER
783 002246 012703 000010 MOV #10,R3 ;SET NO. OF BITS
784 002252 032704 000001 CLP0: BIT #1,F4 ;SEE IF ONE BIT
785 002256 001402 BEQ CLP1 ;IF NOT: BR
786 002260 005267 176542 INCR PARCNT ;RUMP COUNTER
787 002264 000241 CLP1: CLC
788 002266 006004 POR R4 ;ROTATE TO NEXT BIT
789 002270 005303 DEC R3
790 002272 001367 BNE CLP0 ;CONTINUE FOR ALL BITS
791 002274 112104 MOVH (R1)+,R4
792 002276 042704 177400 BIC #177400,P4
793 002302 032767 000001 176516 BIT #1,PARCNT ;SEE IF ODD # OF ONE BITS
794 002310 001002 BNE CLP2 ;IF SO: BR
795 002312 052704 000400 BIS #400,F4 ;SET PARITY BIT
796 002316 000207 CLP2: RTS PC ;EXIT
797
798 002320 010446 XOR: MOV P4,-(SP) ;XOR SUBROUTINE: R4 WITH XORS
799 002322 046716 176466 RIC XORS,(SP)
800 002326 040467 176462 BIC R4,XORS
801 002332 052667 176456 BIS (SP)+,XORS
802 002336 016704 176452 MOV XORS,R4
803 002342 000207 RTS PC
804
805 002344 016767 176454 176442 LPC: MOV ACTLPC,XORS
806 002352 012104 173026 LPC1: MOV (R1)+,R4 ;LOCATION EFFECTED BY SWITCHES
807 002354 022701 CMP #173026,R1 ;IF SO: SKIP LOC. BY BRANCHING
808 002360 001402 BEQ LPC2
809 002362 004767 177732 JSR PC,XOR
810 002366 005300 LPC2: DEC R0
811 002370 001370 BNE LFC1
812 002372 016767 176416 176424 MOV XORS,ACTLPC
813 002400 000207 RTS PC
814
815 002402 104401 TYPROM: TYPE ;TYPE HEADER
816 002404 006023 TYPHDR
817 002406 016700 MOV KOMSA1,R0 ;POINT TO 1ST ROM SPACE
818 002412 016701 176370 MOV DATLN1,P1 ;PUT LENGTH IN R1
819 002416 006201 ASR R1 ;CONVERT TO WORDS
820 002420 001402 BEQ TYPR1 ;BRANCH IF 1ST ROM SPACE NOT USED
821 002422 004767 000026 JSR PC,TYP ;GO TYPE 1ST ADDR. SPACE
822 002426 016700 176356 TYPR1: MOV KOMSA2,R0 ;POINT TO 2ND ADDR. SPACE
823 002432 016701 176354 MOV DATLN2,P1 ;PUT LENGTH IN R1
824 002436 006201 ASR P1 ;CONVERT TO WORDS
825 002440 001402 BEQ ENDOT ;BR IF 2ND ADDR. SPACE NOT USED
826 002442 004767 000006 JSR PC,TYP ;GO TYPE 2ND ADDR. SPACE
827 002446 104401 ENDOT: TYPE
828 002450 005476 EOTST
829 002452 000000 HALT
830
831 002454 104401 TYP: TYPE
832 002456 006015 CARLF

```

```

833 002460 000403 BR TYP3
834 002462 032700 000003 TYP1: BIT #3,R0 ;ADDRESS MULTIPLE OF 4?
835 002466 001006 BNE TYP2 ;IF NOT: BR
836 002470 104401 TYP3: TYPE
837 002472 006015 CARLF
838 002474 010046 MOV R0,-(SP) ;PUT ADDRESS ON STACK
839 002476 104402 TYPOC ;TYPE ADDR.
840 002500 104401 TYPE
841 002502 006060 COLON
842 002504 012046 TYP2: MOV (R0)+,-(SP) ;PUT DATA ON STACK
843 002506 104402 TYPOC ;TYP DATA
844 002510 104401 TYPE ;TYPE 2 SPACES
845 002512 006020 SP2
846 002514 005301 DEC R1 ;FINISHED?
847 002516 001361 BNE TYPA ;IF NOT: BR
848 002520 000207 RTS PC ;RETURN
849 002522 005000 CLR R0
850 002524 062700 000002 AUTACT: CLR ;BUMP TABLE INDEX
851 002530 026027 005226 177777 AUT1: ADD #2,R0 ;CHECKED ALL KNOWN VERSIONS?
852 002536 001425 BEQ AUTERR ;IF SO: BR
853 002540 026067 004644 176254 CMP TXCRC(R0),ACTCRC ;DOES THIS CRC AGREE?
854 002546 001366 BNE AUT1 ;IF NOT: KEEP LOOKING
855 002550 023727 000042 002060 CMP #42,*SENDAD ;UNDER ACT AUTO ACCEPT?
856 002556 001405 BEQ AUT2 ;IF SO: BR
857 002560 016067 005226 000002 MOV TMSG(R0),AUT2 ;SET UP VERSION MESSAGE
858 002566 104401 TYPE
859 002570 000000 AUT2: 0
860 002572 016067 004644 176216 AUT3: MOV TXCRC(R0),EXCRC ;SET EXPECTED CRC
861 002600 016067 004714 176212 MOV TXLPC(R0),LXLPC ;SET EXPECTED LPC
862 002606 000167 177124 JMP CK1 ;CHECK LPC
863
864 002612 104401 AUTERR: TYPE
865 002614 006064 AUTERR:
866 002616 012767 000001 176224 MOV #1,$FATAL ;MOVE TO MAILBOX ERROR NO. **** 1 ****
867 002624 012767 000001 176214 MOV #1,$MSGTYP ;SET MAILBOX FOR FATAL ERROR
868 002632 000000 HALT ;AUTO ACCEPT FAILED
869
870 ;SBTTL TTY INPUT ROUTINE
871
872 ;*****
873 002634 177560 STKB: .WORD 177560 ;TTY KBD STATUS
874 002636 177562 STKB: .WORD 177562 ;TTY KBD BUFFER
875
876 .ENABL LSB
877
878 .DSABL LSB
879
880 ;*****
881 ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
882 ;*CALL:
883 ;* RDCHP ;INPUT A SINGLE CHARACTER FROM THE TTY
884 ;* RETURN HERE ;CHARACTER IS ON THE STACK
885 ;* ;WITH PARITY BIT STRIPPED OFF
886 ;
887
888 002640 011646 BRDCHR: MOV (SP),-(SP) ;PUSH DOWN THE PC

```

```

889 002642 016666 000004 000002      MOV      4(SP),2(SP)      ;;SAVE THE PS
890 002650 105777 177760      TSTB    #8TKS           ;;WAIT FOR
891 002654 100375                BPL     18              ;;A CHARACTER
892 002656 117766 177754 000004      MOVVB   #STKB,4(SP)     ;;READ THE TTY
893 002664 042766 177600 000004      BIC     #'C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
894 002672 026627 000004 000023      CMP     4(SP),#23      ;;IS IT A CONTROL-S?
895 002700 001013                BNE     35              ;;BRANCH IF NO
896 002702 105777 177726      TSTB    #8TKS           ;;WAIT FOR A CHARACTER
897 002706 100375                BPL     25              ;;LOOP UNTIL ITS THERE
898 002710 117746 177722      MOVVB   #STKB,-(SP)     ;;GET CHARACTER
899 002714 042716 177600      BIC     #'C177,(SP)    ;;MAKE IT 7-BIT ASCII
900 002720 022627 000021      CMP     (SP)+,#21      ;;IS IT A CONTROL-Q?
901 002724 001366                BNE     29              ;;IF NOT DISCARD IT
902 002726 000750                BR      19              ;;YES, RESUME
903 002730 026627 000004 000140 301    CMP     4(SP),#140     ;;IS IT UPPER CASE?
904 002736 002407                BLT     45              ;;BRANCH IF YES
905 002740 026627 000004 000175      CMP     4(SP),#175     ;;IS IT A SPECIAL CHAR?
906 002746 003003                BGT     49              ;;BRANCH IF YES
907 002750 042766 000004 000004      BIC     #40,4(SP)     ;;MAKE IT UPPER CASE
908 002756 000002                RTI     49              ;;GO BACK TO USER
909
910
911
912
913
914
915
916 002760 010346                $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
917 002762 005046                CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
918 002764 012703 003214      MOV     #STTYIN,R3     ;;GET ADDRESS
919 002770 022703 003224      CMP     #STTYIN+8,,R3  ;;BUFFER FULL?
920 002774 014456                BLOS   46              ;;BR IF YES
921 002776 104405                PDCHR  (SP)+,(R3)       ;;GO READ ONE CHARACTER FROM THE TTY
922 003000 112613      MOVVB   (SP)+,(R3)     ;;GET CHARACTER
923 003002 122713 000177 1001    CMPB    #177,(R3)      ;;IS IT A RUBOUT
924 003006 001022                BNE     50              ;;RR IF NO
925 003010 005716                TST    (SP)            ;;IS THIS THE FIRST RUBOUT?
926 003012 001007                BNE     60              ;;RR IF NO
927 003014 112767 000134 000170      MOVVB   #' \,96        ;;TYPE A BACK SLASH
928 003022 104401 003212      TYPE    ,96            ;;
929 003026 012716 177777      MOV     #-1,(SP)       ;;SET THE RUBOUT KEY
930 003032 005303                DEC     R3              ;;BACKUP BY ONE
931 003034 020327 003214      CMP     R3,#STTYIN    ;;STACK EMPTY?
932 003040 103434                BLO     48              ;;BR IF YES
933 003042 111367 000144      MOVVB   (R3),96        ;;SETUP TO TYPEOUT THE DELETED CHAR.
934 003046 104401 003212      TYPE    ,96            ;;GO TYPE
935 003052 000746                BR      20              ;;GO READ ANOTHER CHAR.
936 003054 005716                TST    (SP)            ;;RUBOUT KEY SET?
937 003056 001406                BR      70              ;;BR IF NO
938 003060 112767 000134 000124      MOVVB   #' \,96        ;;TYPE A BACK SLASH
939 003066 104401 003212      TYPE    ,96            ;;
940 003072 005016                CLR     (SP)           ;;CLEAR THE RUBOUT KEY
941 003074 122713 000025 701    CMPB    #25,(R3)       ;;IS CHARACTER A CTRL U?
942 003100 001003                BNE     86              ;;BR IF NO
943 003102 104401 003230      TYPE    #CNTLU        ;;TYPE A CONTROL "U"
944 003106 000726                BR      15              ;;GO START OVER

```

```

945 003110 122713 000022 801    CHPR    #22,(R3)       ;;IS CHARACTER A "R"?
946 003114 001011                BNE     38              ;;BRANCH IF NO
947 003116 105013                CLRB   (R3)            ;;CLEAR THE CHARACTER
948 003120 104401 003225      TYPE    #CRLF          ;;TYPE A "CR" & "LF"
949 003124 104401 003214      TYPE    #STTYIN        ;;TYPE THE INPUT STRING
950 003130 000717                BR      28              ;;GO PICKUP ANOTHER CHARACTER
951 003132 104401 003224 401    TYPE    #QUES          ;;TYPE A "?"
952 003136 000712                BR      18              ;;CLEAR THE BUFFER AND LOOP
953 003140 111367 000046 301    MOVVB   (R3),96        ;;ECHO THE CHARACTER
954 003144 104401 003212      TYPE    ,96            ;;
955 003150 122723 000015      CMPB    #15,(R3)+      ;;CHECK FOR RETURN
956 003154 001305                BNE     26              ;;LOOP IF NOT RETURN
957 003156 105063 177777      CLPB   -1(R3)         ;;CLEAR RETURN (THE 15)
958 003162 104401 003226      TYPE    #LF            ;;TYPE A LINE FEED
959 003166 005726                TST    (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
960 003170 012603                MOV     (SP)+,R3       ;;RESTORE R3
961 003172 011646                MOV     (SP),-(SP)     ;;ADJUST THE STACK AND PUT ADDRESS OF THE
962 003174 016666 000004 000002      MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
963 003202 012766 003214 000004      MOV     #STTYIN,4(SP)
964 003210 000002                RTI     96              ;;RETURN
965 003212 000          961    ,BYTE   0              ;;STORAGE FOR ASCII CHAR, TO TYPE
966 003213 000          961    ,BYTE   0              ;;TERMINATOR
967 003214 000010      STTYIN: ,BLKB   0      ;;RESERVE 8 BYTES FOR TTY INPUT
968 003224 077          961    ,ASCIZ  "???"     ;;QUESTION MARK
969 003225 015          961    ,ASCIZ  <15>    ;;CARRIAGE RETURN
970 003226 000012      $LFI:  ,ASCIZ  <12>    ;;LINE FEED
971 003230 052536 000 001    ,ASCIZ  /"U/<15><12>  ;;CONTROL "U"
972 003235 136 000507 000012  ,ASCIZ  /"G/<15><12>  ;;CONTROL "G"
973 003242 005015 053523 020122  $MSWR: ,ASCIZ  <15><12>/$WR = /
974 003250 020075                $MNEW: ,ASCIZ  / NEW = /
975 003253 040 0047040 053505
976 003260 036440 000004
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991 003264 011646                $RDCT: MOV     (SP),-(SP)   ;;PROVIDE SPACE FOR THE
992 003266 016666 000004 000002      MOV     4(SP),2(SP)   ;;INPUT NUMBER
993 003274 010046                MOV     R0,-(SP)      ;;PUSH R0 ON STACK
994 003276 010146                MOV     R1,-(SP)      ;;PUSH R1 ON STACK
995 003300 010246                MOV     R2,-(SP)      ;;PUSH R2 ON STACK
996 003302 104406 101    RDCLIN  (SP)+,R0       ;;READ AN ASCII LINE
997 003304 012600      MOV     R0,R0         ;;GET ADDRESS OF 1ST CHARACTER
998 003306 010067 000100      CLR     R0,50         ;;AND SAVE IT
999 003312 005001      CLR     R1            ;;CLEAR DATA WORD
1000 003314 005002      CLR     R2

```

1041 003316 112046 26: MOVB (R0)+,(SP) ;;PICKUP THIS CHARACTER
1042 003320 001420 BEQ 36 ;;IF ZERO GET OUT
1043 003322 122716 000060 CMPR #'0,(SP) ;;MAKE SURE THIS CHARACTER
1044 003326 003026 FGT 48 ;;IS AN OCTAL DIGIT
1045 003330 122716 000067 CMPB #'7,(SP)
1046 003334 002423 RLT 48
1047 003336 006301 ASL R1 ;;*2
1048 003340 006102 ROL R2
1049 003342 006301 ASL R1 ;;*4
1050 003344 006102 ROL R2
1051 003346 006301 ASL R1 ;;*8
1052 003350 006102 ROL R2
1053 003352 042716 17770 BIC #'C7,(SP) ;;STRIP THE ASCII JUNK
1054 003356 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
1055 003360 000756 BR 26 ;;LOOP
1056 003362 005726 30: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
1057 003364 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
1058 003370 010267 000026 MOV R2,#SHIOCT
1059 003374 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
1060 003376 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
1061 003400 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
1062 003402 000002 RTI ;;RETURN
1063 003404 005726 40: TST (SP)+ ;;CLEAN PARTIAL FROM STACK
1064 003406 105010 CLMB (R0) ;;SET A TERMINATOR
1065 003410 104401 TYPE ;;TYPE UP THRU THE BAD CHAR,
1066 003412 000000 50: WORD 0 ;;WORD
1067 003414 104401 TYPE ,SQUES ;;"?" "CR" & "LF"
1068 003420 000730 BR 13 ;;TRY AGAIN
1069 003422 000000 SHIOCT: WORD 0 ;;HIGH ORDER BITS GO HERE
1070 003424 SBTTL TYPE ROUTINE
1071 003426 ;;*****
1072 003428 ;;ROUTINE TO TYPE ASCII MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE,
1073 003430 ;;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED,
1074 003432 ;;NOTE1: #NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER,
1075 003434 ;;NOTE2: #FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED,
1076 003436 ;;NOTE3: #FILLC CONTAINS THE CHARACTER TO FILL AFTER,
1077 003438 ;;
1078 003440 ;;*CALL:
1079 003442 ;;*1) USING A TRAP INSTRUCTION
1080 003444 ;;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCII STRING
1081 003446 ;;*OR
1082 003448 ;;* TYPE
1083 003450 ;;* MESADP
1084 003452 ;;
1085 003454 ;;
1086 003456 ;;
1087 003458 6TYPE: TSTR \$TPFLG ;;IS THERE A TERMINAL?
1088 003460 18 RPL 18 ;;R IF YES
1089 003462 000000 HALT ;;HALT HERE IF NO TERMINAL
1090 003464 000430 BR 38 ;;LEAVE
1091 003466 010046 10: MOV R0,-(SP) ;;SAVE R0
1092 003468 017600 000002 MOV 02(SP),R0 ;;GET ADDRESS OF ASCII STRING
1093 003470 122767 000001 175414 CMPB #APTENV,\$ENV ;;RUNNING IN APT MODE
1094 003472 001011 BNE 626 ;;NO,GO CHECK FOR APT CONSOLE
1095 003474 132767 000100 175405 BITB #APTSPOOL,\$ENVM ;;SPOOL MESSAGE TO APT
1096 003476 001405 BEQ 626 ;;NO,GO CHECK FOR CONSOLE

1057 003464 010067 000004 MOV R0,616 ;;SETUP MESSAGE ADDRESS FOR APT
1058 003470 004767 000230 JSR PC,\$ATY3 ;;SPOOL MESSAGE TO APT
1059 003474 000000 610: WORD 0 ;;MESSAGE ADDRESS
1060 003476 132767 000040 175363 626: PTTB ;;APT CONSOLE SUPPRESSED
1061 003504 001003 BNF 60S ;;YES,SKIP TYPE OUT
1062 003506 112046 26: MOVB (R0)+,(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1063 003510 001005 RNE 46 ;;BR IF IT ISN'T THE TERMINATOR
1064 003512 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
1065 003514 012600 60S: MOV (SP)+,R0 ;;RESTORE R0
1066 003516 062716 36: ADD #7,(SP) ;;ADJUST RETURN PC
1067 003522 000002 RTI ;;RETURN
1068 003524 122716 40: CMPB #'T,(SP) ;;BRANCH IF <HT>
1069 003530 001430 BEQ 86 ;;BRANCH IF NOT <CRLF>
1070 003532 122716 000200 CMPR #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
1071 003536 001006 RNE 56 ;;POP <CR><LF> EQUIV
1072 003540 005726 TST (SP)+ ;;TYPE A CR AND LF
1073 003542 104401 TYPE ;;TYPE A CR AND LF
1074 003544 003225 SCRLF
1075 003546 105067 000130 CLRR \$CHARCNT ;;CLEAN CHARACTER COUNT
1076 003552 000755 BR 26 ;;GET NEXT CHARACTER
1077 003554 004767 000056 50: JSR PC,\$TYFEC ;;GO TYPE THIS CHARACTER
1078 003560 126726 000130 60: CMPE #FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
1079 003564 001350 BNF 26 ;;IF NO GO GET NEXT CHAR,
1080 003566 016746 000120 MOV #NULL,-(SP) ;;GET # OF FILLER CHARS, NEEDED
1081 003572 105366 000001 70: DECB 1(SP) ;;AND THE NULL CHAR,
1082 003576 002770 BLT 68 ;;DOES A NULL NEED TO BE TYPED?
1083 003600 004767 000032 JSR PC,\$TYPEC ;;BR IF NO--GO POP THE NULL OFF OF STACK
1084 003604 105367 000072 DECB \$CHARCNT ;;GO TYPE A NULL
1085 003610 000770 BR 76 ;;DO NOT COUNT AS A COUNT
1086 ;;LOOP
1087 ;;
1088 ;;
1089 ;;
1090 ;;HORIZONTAL TAB PROCESSOR
1091 003612 112716 000040 80: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
1092 003616 004767 000014 90: JSR PC,\$TYPEC ;;TYPE A SPACE
1093 003622 132767 000007 000052 BITB #7,\$CHARCNT ;;BRANCH IF NOT AT
1094 003630 001372 RNE 98 ;;TAB STOP
1095 003632 005726 TST (SP)+ ;;POP SPACE OFF STACK
1096 003634 000724 BR 26 ;;GET NEXT CHARACTER
1097 003636 105777 000044 6TYPEC: TSTB \$STPB ;;WAIT UNTIL PRINTER IS READY
1098 003642 100375 BPL \$TYPEC ;;LOAD CHAR TO BE TYPED INTO DATA REG,
1099 003644 116677 000002 000036 MOVB 2(SP),#STPB ;;IS CHARACTER A CARRIAGE RETURN?
1100 003652 122766 000015 000002 CMPE #CR,2(SP) ;;BRANCH IF NO
1101 003660 001003 BNE 18 ;;YES--CLEAR CHARACTER COUNT
1102 003662 105067 000014 CLRR \$CHARCNT ;;EXIT
1103 003666 000406 BR \$TYPEC ;;IS CHARACTER A LINE FEED?
1104 003670 122766 000012 000002 10: CMPB #LF,2(SP) ;;BRANCH IF YES
1105 003676 001402 BEQ \$TYPEC ;;COUNT THE CHARACTER
1106 003700 105227 INCB (PC)+ ;;CHARACTER COUNT STORAGE
1107 003702 000000 \$CHARCNT,WORD 0
1108 003704 000207 \$TYPEC: PTS PC
1109 003706 177564 \$TPS: WORD 177564 ;;TTY PRINTER STATUS REG. ADDRESS
1110 003710 177566 \$TPB: WORD 177566 ;;TTY PRINTER BUFFER REG. ADDRESS
1111 003712 000 \$NULL: BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
1112 003714 002 \$FILLS: BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED

```

1113 003714 012          $FILLC1 ,BYTE 12          ;;INSERT FILL CHARS, AFTER A "LINE FEED"
1114 003715 000          $TPFLG1 ,BYTE 0           ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1115                                     ,SBTTL APT COMMUNICATIONS ROUTINE
1116
1117 ;;*****
1118 003716 112767 000001 000236 $ATY11 MOVF #1,$FFLG          ;;TO REPORT FATAL ERROR
1119 003724 112767 000001 000226 $ATY13 MOVFB #1,$MFLG        ;;TO TYPE A MESSAGE
1120 003732 000403          BR $ATYC
1121 003734 112767 000001 000220 $ATY41 MOVFB #1,$FFLG          ;;TO ONLY REPORT FATAL ERROR
1122 003742          $ATYC1
1123 003742 010046          MOV R0,-(SP)          ;;PUSH R0 ON STACK
1124 003744 010146          MOV R1,-(SP)          ;;PUSH R1 ON STACK
1125 003746 105767 000206          TSTB $MFLG          ;;SHOULD TYPE A MESSAGE?
1126 003752 001450          BEQ 58          ;;IF NOT: BR
1127 003754 122767 000001 175104 CMPB #APTENV,$ENV          ;;OPERATING UNDER APT?
1128 003762 001031          BNE 38          ;;IF NOT: BR
1129 003764 132767 000100 175075 BITB #APTSPOOL,$ENVM          ;;SHOULD SPOOL MESSAGES?
1130 003772 001425          BFO 38          ;;IF NOT: BR
1131 003774 017600 000004          MOV 04(SP),R0          ;;GET MESSAGE ADDR.
1132 004000 062766 000002 000004 ADD #2,4(SP)          ;;BUMP RETURN ADDR.
1133 004006 005767 175034          18: TST $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
1134 004012 001375          BNE 18          ;;IF NOT: WAIT
1135 004014 010067 175042          MOV R0,$MSGAD          ;;PUT ADDR IN MAILBOX
1136 004020 105720          28: TSTB (R0)+          ;;FIND END OF MESSAGE
1137 004022 001376          BRL 28
1138 004024 166700 175032          SUB $MSGAD,R0          ;;SUB START OF MESSAGE
1139 004030 006200          ASR R0          ;;GET MESSAGE LGTH IN WORDS
1140 004032 010067 175026          MOV R0,$MSGLGT          ;;PUT LENGTH IN MAILBOX
1141 004036 012767 000004 175002 MOV #4,$MSGTYPE          ;;TELL APT TO TAKE MSG.
1142 004044 000413          BR 58
1143 004046 017667 000004 000016 36: MOV 04(SP),46          ;;PUT MSG ADDR IN JSR LINKAGE
1144 004054 062766 000002 000004 ADD #2,4(SP)          ;;BUMP RETURN ADDRESS
1145 004062 016746 173710          MOV 177776,-(SP)          ;;PUSH 177776 ON STACK
1146 004066 004767 177332          JSR PC,$TYPE          ;;CALL TYPE MACRO
1147 004072 000000          48: ,WORD 0
1148 004074          58:
1149 004074 105767 000062          108: TSTB $FFLG          ;;SHOULD REPORT FATAL ERROR?
1150 004100 001416          BEQ 128          ;;IF NOT: BR
1151 004102 005767 174760          TST $ENV          ;;RUNNING UNDER APT?
1152 004106 001413          BEQ 126          ;;IF NOT: BR
1153 004110 005767 174732          118: TST $MSGTYPE          ;;FINISHED LAST MESSAGE?
1154 004114 001375          BNE 118          ;;IF NOT: WAIT
1155 004116 017667 000004 174724 MOV 04(SP),$FATAL          ;;GET ERROR #
1156 004124 062766 000002 000004 ADD #2,4(SP)          ;;BUMP RETURN ADDR.
1157 004132 005267 174710          INC $MSGTYPE          ;;TELL APT TO TAKE ERROR
1158 004136 105067 000002          128: CLRFB $FFLG          ;;CLEAR FATAL FLAG
1159 004142 105067 000013          CLRFB $LFLG          ;;CLEAR LOG FLAG
1160 004146 105067 000006          CLRFB $MFLG          ;;CLEAR MESSAGE FLAG
1161 004152 012601          MOV (SP)+,R1          ;;POP STACK INTO R1
1162 004154 012600          MOV (SP)+,R0          ;;POP STACK INTO R0
1163 004156 000207          RTS PC          ;;RETURN
1164 004160 000          $MFLG1 ,BYTE 0           ;;MESSG. FLAG
1165 004161 000          $LFLG1 ,BYTE 0           ;;LOG FLAG
1166 004162 000          $FFLG1 ,BYTE 0           ;;FATAL FLAG
1167 004164          ,EVEN
1168 000200          APTISIZE=200

```

```

1169 000001          APTENV=001
1170 000100          APTSPOOL=100
1171 000040          APTCSUP=040
1172                                     ,SBTTL BINARY TO OCTAL (ASCII) AND TYPE
1173
1174 ;;*****
1175 ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1176 ;;OCTAL (ASCII) NUMBER AND TYPE IT.
1177 ;;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1178 ;;$CALL1
1179 ;;      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
1180 ;;      TYPOS          ;;CALL FOR TYPEOUT
1181 ;;      ,BYTE 11          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1182 ;;      ,BYTE M          ;;M=1 OR 0
1183 ;;                                     ;;1=TYPE LEADING ZEROS
1184 ;;                                     ;;0=SUPPRESS LEADING ZEROS
1185 ;;
1186 ;;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1187 ;;$TYPOS OR $TYPOC
1188 ;;$CALL1
1189 ;;      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
1190 ;;      TYPON          ;;CALL FOR TYPEOUT
1191 ;;
1192 ;;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1193 ;;$CALL1
1194 ;;      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
1195 ;;      TYPOC          ;;CALL FOR TYPEOUT
1196 ;;
1197 004164 017646 000000          $TYPOS1 MOV 0(SP),-(SP)          ;;PICKUP THE MODE
1198 004170 116667 000001 000211 MOVB 1(SP),$0FILL          ;;LOAD ZERO FILL SWITCH
1199 004176 112667 000207          MOVFB (SP)+,$0MODE+1          ;;NUMBER OF DIGITS TO TYPE
1200 004202 062716 000002          ADD #2,(SP)          ;;ADJUST RETURN ADDRESS
1201 004206 000406          BR $TYPON
1202 004210 112767 000001 000171 $TYPOC1 MOVFB #1,$0FILL          ;;SET THE ZERO FILL SWITCH
1203 004216 112767 000006 000165          MOVFB #6,$0MODE+1          ;;SET FOR SIX(6) DIGITS
1204 004224 112767 000005 000154 $TYPON1 MOVFB #5,$0CNT          ;;SET THE ITERATION COUNT
1205 004232 010346          MOV R3,-(SP)          ;;SAVE R3
1206 004234 010446          MOV R4,-(SP)          ;;SAVE R4
1207 004236 010546          MOV R5,-(SP)          ;;SAVE R5
1208 004240 116704 000145          MOVFB $0MODE+1,R4          ;;GET THE NUMBER OF DIGITS TO TYPE
1209 004244 005404          NEG R4
1210 004246 062704 000006          ADD #6,R4          ;;SUBTRACT IT FOR MAX, ALLOWED
1211 004252 110467 000132          MOVFB R4,$0MODE          ;;SAVE IT FOR USE
1212 004256 116704 000125          MOVFB $0FILL,R4          ;;GET THE ZERO FILL SWITCH
1213 004262 016605 000012          MOV 12(SP),R5          ;;PICKUP THE INPUT NUMBER
1214 004266 005003          CLR R3          ;;CLEAR THE OUTPUT WORD
1215 004270 006105          18: ROL R5          ;;ROTATE MSB INTO "C"
1216 004272 000404          BR 38          ;;GO DO MSB
1217 004274 006105          28: ROL R5          ;;FORM THIS DIGIT
1218 004276 006105          ROL R5
1219 004300 006105          ROL R5
1220 004302 010503          MOV R5,R3
1221 004304 006103          38: ROL R3          ;;GET LSB OF THIS DIGIT
1222 004306 105367 000076          DECB $0MODE          ;;TYPE THIS DIGIT?
1223 004312 100016          BPL 78          ;;BR IF NO
1224 004314 042703 177770          RIC #177770,R3          ;;GET RID OF JUNK

```



```
1225 004320 001002 BNE 48 ;TEST FOR 0
1226 004322 005704 TST R4 ;SUPPRESS THIS 0?
1227 004324 001403 REO 56 ;BR IF YES
1228 004326 005204 46: INC R4 ;DON'T SUPPRESS ANYMORE 0'S
1229 004330 005273 000060 BIS #0,R3 ;MAKE THIS DIGIT ASCII
1230 004334 005273 000040 56: BIS #*,R3 ;MAKE ASCII IF NOT ALREADY
1231 004340 110367 000040 MOV# R3,R5 ;SAVE FOR TYPING
1232 004344 104401 004404 TYPE ,R5 ;GO TYPE THIS DIGIT
1233 004350 105367 000032 75: DECR 60CNT ;COUNT BY 1
1234 004354 003347 RGT 26 ;BR IF MORE TO DO
1235 004356 002402 BLT 68 ;BR IF DONE
1236 004360 005204 INC R4 ;INSURE LAST DIGIT ISN'T A BLANK
1237 004362 000744 RF 28 ;GO DO THE LAST DIGIT
1238 004364 012605 66: MOV (SP)+,R5 ;RESTORE R5
1239 004366 012604 MOV (SP)+,R4 ;RESTORE R4
1240 004370 012603 MOV (SP)+,R3 ;RESTORE R3
1241 004372 016666 000002 000004 MOV 2(SP),4(SP) ;SET THE STACK FOR RETURNING
1242 004400 012616 MOV (SP)+,(SP)
1243 004402 000002 RTI ;RETURN
1244 004404 0000 ;STORAGE FOR ASCII DIGIT
1245 004405 0000 ;TERMINATOR FOR TYPE ROUTINE
1246 004406 0000 SOCNT: ;OCTAL DIGIT COUNTER
1247 004407 0000 $FILL: ;ZERO FILL SWITCH
1248 004410 000000 $MODE: ;NUMBER OF DIGITS TO TYPE
1249 SBTTL POWER DOWN AND UP ROUTINES
1250
1251 ;*****
1252 ;POWER DOWN ROUTINE
1253 004412 012737 004552 000024 $PWRDN: MOV #FILLUP,#PWRVEC ;SET FOR FAST UP
1254 004420 012737 000340 000026 MOV #340,#PWRVEC+2 ;PPIO:7
1255 004426 010046 MOV R0,-(SP) ;PUSH R0 ON STACK
1256 004430 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
1257 004432 010246 MOV R2,-(SP) ;PUSH R2 ON STACK
1258 004434 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
1259 004436 010446 MOV R4,-(SP) ;PUSH R4 ON STACK
1260 004440 010546 MOV R5,-(SP) ;PUSH R5 ON STACK
1261 004442 017746 174332 MOV $SWR,-(SP) ;PUSH $SWR ON STACK
1262 004446 010667 000104 MOV SP,$SAVR6 ;SAVE SP
1263 004452 012737 004464 000024 MOV $PWRIP,#PWRVEC ;SET UP VECTOR
1264 004460 000000 HALT
1265 004462 000776 BR ,-2 ;HANG UP
1266
1267 ;*****
1268 ;POWER UP ROUTINE
1269 004464 012737 004552 000024 $PWRUP: MOV #FILLUP,#PWRVEC ;SET FOR FAST DOWN
1270 004472 016706 000060 MOV $SAVR6,SP ;GET SP
1271 004476 005067 000054 CLR $SAVR6 ;WAIT LOOP FOR THE TTY
1272 004502 005267 000050 16: INC $SAVR6 ;WAIT FOR THE INC
1273 004506 001375 BNE 18 ;OF WORD
1274 004510 012677 174264 MOV (SP)+,$SWR ;POP STACK INTO $SWR
1275 004514 012605 MOV (SP)+,R5 ;POP STACK INTO R5
1276 004516 012604 MOV (SP)+,R4 ;POP STACK INTO R4
1277 004520 012603 MOV (SP)+,R3 ;POP STACK INTO R3
1278 004522 012602 MOV (SP)+,R2 ;POP STACK INTO R2
1279 004524 012601 MOV (SP)+,R1 ;POP STACK INTO R1
1280 004526 012600 MOV (SP)+,R0 ;POP STACK INTO R6
```

```
1281 004530 012737 004412 000024 MOV $PWRDN,#PWRVEC ;SET UP THE POWER DOWN VECTOR
1282 004536 012737 000340 000026 MOV #340,#PWRVEC+2 ;PPIO:7
1283 004544 104401 TYPE ;REPORT THE POWER FAILURE
1284 004546 004560 $PWRMG: ;WORD $POWER ;POWER FAIL MESSAGE POINTER
1285 004550 000002 PTI
1286 004552 000000 $ILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
1287 004554 000776 BR ,-2 ;BEFORE THE POWER DOWN WAS COMPLETE
1288 004556 000000 $SAVR6: 0 ;PUT THE SP HERE
1289 004560 005015 047520 042527 $POWER: ;ASCIZ <15><12>"POWER"
1290 004566 000122 .EVEN
1291 SBTTL TRAP DECODER
1292
1293 ;*****
1294 ;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1295 ;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1296 ;OF THE DESIRED ROUTINE, THEN USING THE ADDRESS OBTAINED IT WILL
1297 ;GO TO THAT ROUTINE.
1298
1299 1300 004570 010046 $TRAP: MOV R0,-(SP) ;SAVE R0
1301 004572 016600 000002 MOV 2(SP),R0 ;GET TRAP ADDRESS
1302 004576 005740 TST -(R0) ;BACKUP BY 2
1303 004600 111000 MOV# (R0),R0 ;GET RIGHT BYTE OF TRAP
1304 004602 006301 ASL R0 ;POSITION FOR INDEXING
1305 004604 016000 004621 MOV $TRPAD(R0),R0 ;INDEX TO TABLE
1306 004610 000200 FIS R0 ;GO TO ROUTINE
1307
1308 ;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1309
1310 1311 004612 011646 $TRAP2: MOV (SP),-(SP) ;MOVE THE PC DOWN
1312 004614 016666 000004 000002 MOV 4(SP),2(SP) ;MOVE THE PSW DOWN
1313 004622 000002 RTI ;RESTORE THE PSW
1314
1315 SBTTL TRAP TABLE
1316
1317 ;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1318 ;BY THE "TRAP" INSTRUCTION.
1319
1320 ; ROUTINE
1321 ; -----
1322 004624 004612 $TRPAD: ;WORD $TRAP2
1323 004626 003424 $TYPE ;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
1324 004630 004210 $TYPOC ;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1325 004632 004164 $TYPOS ;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
1326 004634 004224 $TYPON ;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
1327
1328 1329 004636 002640 $RDCHR ;CALL=RDCHR TRAP+5(104405) TTY TYPEIN CHARACTER ROUTINE
1330 004640 002760 $RDLIN ;CALL=RDLIN TRAP+6(104406) TTY TYPEIN STRING ROUTINE
1331 004642 003264 $RDOCT ;CALL=RDOCT TRAP+7(104407) READ AN OCTAL NUMBER FROM TTY
1332
1333 TXCRC: -1
1334 004646 000571 571 ;M9301 = YA VERSION
1335 004650 000457 457 ;M9301 = YB VERSION
1336 004652 000243 243 ;M9301 = YC VERSION
```

1337	004654	000635	635	JM9400 = YA(OR YC) VERSION
1338	004656	000207	207	JM9301 = YF VERSION
1339	004660	000670	670	JM7942 = YB VERSION
1340	004662	000132	132	JM9301 = YD VERSION
1341	004664	000374	374	JM9400 = YH (OR YK) VERSION
1342	004666	000630	630	JM9311 VERSION = REVISION * (ORIGINAL REV.)
1343	004670	000536	536	JM9301 = YH VERSION
1344	004672	000752	752	JM9301 = YE VERSION
1345	004674	000633	633	JM9301 = YJ VERSION
1346	004676	000650	650	JM9400 = YN VERSION
1347	004700	000710	710	JM9311 VERSION = REVISION A (SECOND REV.)
1348	004702	177777	-1	
1349	004704	177777	-1	
1350	004706	177777	-1	
1351	004710	177777	-1	
1352	004712	177777	-1	
1353				
1354	004714	177777		
1355	004716	133725	133725	JM9301 = YA VERSION
1356	004720	017563	17563	JM9301 = YB VERSION
1357	004722	141744	141744	JM9301 = YC VERSION
1358	004724	047613	47613	JM9400 = YA(OR YC) VERSION
1359	004726	114175	114175	JM9301 = YF VERSION
1360	004730	146126	146126	JM7942 = YB VERSION
1361	004732	132161	132161	JM9301 = YD VERSION
1362	004734	143466	143466	JM9400 = YH(OR YK) VERSION
1363	004736	036751	36751	JM9311 VERSION = REVISION * (ORIGINAL REV.)
1364	004740	125411	125411	JM9301 = YH VERSION
1365	004742	066246	066246	JM9301 = YE VERSION
1366	004744	132367	132367	JM9301 = YJ VERSION
1367	004746	030210	30210	JM9400 = YN VERSION
1368	004750	036743	36743	JM9311 VERSION = REVISION A (SECOND REV.)
1369	004752	177777	-1	
1370	004754	177777	-1	
1371	004756	177777	-1	
1372	004760	177777	-1	
1373	004762	177777	-1	
1374	004764	177777	-1	
1375				
1376	004766	177777		
1377	004770	001000	1000	JM9301 = YA VERSION
1378	004772	001000	1000	JM9301 = YB VERSION
1379	004774	001000	1000	JM9301 = YC VERSION
1380	004776	001000	1000	JM9400 = YA(OR YC) VERSION
1381	005000	001000	1000	JM9301 = YF VERSION
1382	005002	004000	4000	JM7942 = YB VERSION
1383	005004	001000	1000	JM9301 = YD VERSION
1384	005006	001000	1000	JM9400 = YH(OR YK) VERSION
1385	005010	001000	1000	JM9311 VERSION REVISION * (ORIGINAL VERSION)
1386	005012	000734	734	JM9301 = YH VERSION
1387	005014	001000	1000	JM9301 = YE VERSION
1388	005016	001000	1000	JM9301 = YJ VERSION
1389	005020	001000	1000	JM9400 = YN VERSION
1390	005022	001000	1000	JM9311 VERSION = REVISION A (SECOND VERSION)
1391	005024	177777	-1	
1392	005026	177777	-1	

1393	005030	177777	-1	
1394	005032	177777	-1	
1395	005034	177777	-1	
1396				
1397	005036	177777		
1398	005040	173000	173000	JM9301 = YA VERSION
1399	005042	173000	173000	JM9301 = YB VERSION
1400	005044	173000	173000	JM9301 = YC VERSION
1401	005046	173000	173000	JM9400 = YA(OR YC) VERSION
1402	005050	173000	173000	JM9301 = YF VERSION
1403	005052	170000	170000	JM7942 = YB VERSION
1404	005054	173000	173000	JM9301 = YD VERSION
1405	005056	173000	173000	JM9400 = YH(OR YK) VERSION
1406	005060	163000	163000	JM9311 VERSION REVISION * (ORIGINAL VERSION)
1407	005062	173000	173000	JM9301 = YH VERSION
1408	005064	173000	173000	JM9301 = YE VERSION
1409	005066	173000	173000	JM9301 = YJ VERSION
1410	005070	173000	173000	JM9400 = YN VERSION
1411	005072	163000	163000	JM9311 VERSION REVISION A (SECOND VERSION)
1412	005074	177777	-1	
1413	005076	177777	-1	
1414	005100	177777	-1	
1415	005102	177777	-1	
1416	005104	177777	-1	
1417				
1418	005106	177777		
1419	005110	001000	1000	JM9301 = YA VERSION
1420	005112	001000	1000	JM9301 = YB VERSION
1421	005114	001000	1000	JM9301 = YC VERSION
1422	005116	001000	1000	JM9400 = YA(OR YC) VERSION
1423	005120	001000	1000	JM9301 = YF VERSION
1424	005122	000000	0	JM7942 = YB VERSION
1425	005124	001000	1000	JM9301 = YD VERSION
1426	005126	001000	1000	JM9400 = YH(OR YK) VERSION
1427	005130	001000	1000	JM9311 VERSION REVISION * (ORIGINAL REV.)
1428	005132	000764	764	JM9301 = YH VERSION
1429	005134	001000	1000	JM9301 = YE VERSION
1430	005136	001000	1000	JM9301 = YJ VERSION
1431	005140	001000	1000	JM9400 = YN VERSION
1432	005142	001000	1000	JM9311 VERSION REVISION A (SECOND REV.)
1433	005144	177777	-1	
1434	005146	177777	-1	
1435	005150	177777	-1	
1436	005152	177777	-1	
1437	005154	177777	-1	
1438				
1439	005156	177777		
1440	005160	165000	165000	JM9301 = YA VERSION
1441	005162	165000	165000	JM9301 = YB VERSION
1442	005164	165000	165000	JM9301 = YC VERSION
1443	005166	165000	165000	JM9400 = YA(OR YC) VERSION
1444	005170	165000	165000	JM9301 = YF VERSION
1445	005172	000000	0	JM7942 = YB VERSION
1446				
1447	005174	165000	165000	JM9301 = YD VERSION
1448	005176	165000	165000	JM9400 = YH(OR YK) VERSION

AUTERR	002612	852	864*						
AUT1	002524	850*	854						
AUT2	002570	857*	859*						
AUT3	002572	856	860*						
AVECT1	000000	569							
AVECT2	000000	569							
BIT0	000001	485*							
BIT00	000001	475*	485						
BIT01	000002	474*	484						
BIT02	000004	473*	483						
BIT03	000010	477*	482						
BIT04	000020	471*	481						
BIT05	000040	470*	480						
BIT06	000100	469*	479						
BIT07	000200	468*	478						
BIT08	000400	467*	477						
BIT09	001000	466*	476						
BIT1	000002	484*							
BIT10	002000	465*							
BIT11	004000	464*							
BIT12	010000	463*							
BIT13	020000	462*							
BIT14	040000	461*							
BIT15	100000	460*							
BIT2	000004	483*							
BIT3	000010	482*							
BIT4	000020	481*							
BIT5	000040	480*							
BIT6	000100	479*							
BIT7	000200	478*							
BIT8	000400	477*							
BIT9	001000	476*							
BPTVEC	000014	492*							
CARLF	006015	832	837	1548*					
CHECK	001510	642	644	646	673*				
CH0	001566	679	686*						
CH1	001644	688	697	699*					
CK1	001736	700	713	718*	862				
CK2	002030	719	731	736*					
CLLAST	002204	773	775*						
CLP0	002252	784*	790						
CLP1	002264	785	787*						
CLP2	002316	794	796*						
CL0	002102	751*	757	774					
CL1	002146	765*							
CL2	002170	762	771*						
CL3	002122	753	758*						
COLON	006060	841	1555*						
CR	000015	400*	1099	1109					
CRC	002074	681	689	750*					
CRLF	000200	401*	1070	1109					
DATLN1	001006	523*	638*	664*	678	683	818		
DATLN2	001012	525*	640*	672*	687	691	823		
DDISP	177570	407*	609						
DISPLA	001002	521*	609*	617*					
DISPRE	000174	508*	617						

DS*H	177570	406*	608							
EMTVEC	000030	495*								
ENDOT	002446	825	827*							
EOTST	005476	739	928	1509*						
ERRVEC	000004	488*	606	607*	618*					
EXCRC	001016	527*	637*	652*	699	704	860*			
EXCRNG	005361	703	1493*							
EXLPC	001020	528*	636*	656*	718	722	861*			
EXLPNG	005404	721	1497*							
GETCRC	005311	650	1485*							
GETIN	001370	634	643*							
GETLPC	005335	654	1489*							
GET2	001440	648	657*							
GNS	***** U	507	1323	1324	1325	1326	1329	1330	1331	
GORGIN	002070	742	748*							
HT	000011	398*	1068	1109						
IOTVEC	000020	493*								
LF	000012	399*	1103	1109						
LPC	002344	805	693	805*						
LPC1	002352	806*	811							
LPC2	002366	808	810*							
MSG1	006106	1462	1560*							
MSG10	006244	1469	1581*							
MSG11	006270	1470	1585*							
MSG12	006347	1471	1593*							
MSG13	006364	1472	1596*							
MSG14	006401	1473	1599*							
MSG15	006416	1474	1602*							
MSG16	006433	1475	1605*							
MSG2	006123	1463	1563*							
MSG3	006140	1464	1566*							
MSG4	006155	1465	1569*							
MSG5	006175	1466	1572*							
MSG6	006212	1467	1575*							
MSG7	006227	1468	1578*							
PARCNT	001026	531*	782*	786*	793					
PARITY	002242	758	782*							
PIPO	177772	405*								
PIROVE	000240	499*								
PR0	000000	422*								
PR1	000040	423*								
PR2	000100	424*								
PR3	000140	425*								
PR4	000200	426*								
PR5	000240	427*								
PR6	000300	428*								
PR7	000340	429*								
PS	177776	402*	403	516*						
PSW	177776	403*								
PWRVEC	000024	494*	602*	603*	1253*	1254*	1263*	1269*	1281*	1282*
RDCMR	104405	921	1329*							
RDLIN	104406	906	1330*							
RDOCT	104407	651	655	659	663	667	671	1331*		
RESTRT	001274	626	630*	748						
RESVEC	000010	489*								
ROMSA1	001004	522*	639*	660*	680	682	817			

