

M9312

BOOTSTRAP TERMINATOR 8K
CZM9BA0

AH-E058A-MC

COPYRIGHT © 1978

FICHE 1 OF 1

APR 1978

digital

MADE IN USA

This microfiche card contains a grid of frames, likely representing a data table or a series of small diagrams. The frames are arranged in approximately 12 rows and 4 columns. Each frame contains text and possibly small graphical elements, but the resolution is too low to read the specific content. The frames appear to be organized into sections, with some larger frames in the top-left and bottom-left corners, and smaller, more uniform frames in the middle and right sections.

BO1

NR+ZM988550 PAGE 1 00010000
CZM9BA.F11 13-MAR-78 08:58

780330

PDP10 411

CZM9BA0 M9312 BOOT TERMR 8K

MACY11 30A(1052) 13-
SEQ 0001

.REM !

IDENTIFICATION

PRODUCT CODE: AC-E057A-MC
PRODUCT NAME: CZM9BA0 M9312.BOOT.TERMR.8K
DATE: MARCH, 1978
MAINTAINER: DIAGNOSTIC GROUP

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 SUED 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) 1978 BY DIGITAL EQUIPEMNT CORPORATION

1.0 ABSTRACT

THIS PROGRAM VERIFIES THE ROM INFORMATION FOR THE M9312 BOOTSTRAP TERMINATOR. IT HAS TWO MODES OF OPERATION; STAND-ALONE MODE WHICH REQUIRES OPERATOR INTERVENTION AND APT-MODE.

2.0 REQUIREMENTS

2.1 HARDWARE

ANY PDP-11 UNIBUS PROCESSOR WITH CONSOLE TERMINAL AND/OR HARDWARE SWITCH REGISTER
M9312 BOOT STRAP TERMINATOR
4K MEMORY

2.2 SOFTWARE

THIS PROGRAM REQUIRES THAT THE CORRECT OPERATION OF THE PROCESSOR, MEMORY AND CONSOLE TERMINAL HAVE BEEN VERIFIED BY THE APPROPRIATE DIAGNOSTICS.

3.0 LOADING AND STARTING PROCEDURES

3.1 LOAD THE PROGRAM BY ANY OF THE STANDARD PROCEDURES FOR ABSOLUTE PROGRAM FORMATS.

3.2 STARTING ADDRESS

200- DO CRC VERIFICATION AND SIZING

3.3 RESTART ADDRESS

204- RESTART WITHOUT SIZING

3.4 SPECIAL ENVIRONMENTS

THIS PROGRAM IS APT COMPATIBLE SEE SECTION FOR ETABLE SET-UP.
FOLLOW STANDARD APT PROCEDURES FOR LOADING AND STARTING.

4.0 OPERATING PROCEDURES

4.1 OPERATIONAL SWITCH SETTINGS

THE PROGRAM IS DESIGNED TO USE THE HARDWARE SWITCH REGISTER, HOWEVER IF THIS REGISTER IS NOT AVAILABLE THE PPROGRAM WILL USE LOCATION 176 AS THE SWITCH REGISTER.

SW 15=1 OR UP-- HALT ON ERROR
SW 13=1 OR UP-- INHIBIT ERROR TYPEOUTS
SW 10=1 OR UP-- BELL ON ERROR

4.2 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON THE CONSOLE TERMINAL DURING THE FIRST PASS. ALL OTHER PASSES TAKE LESS THEN 1 SECOND.

4.3 APT PROCEDURES

THERE ARE TWO CHOICES WHEN RUNNING UNDER APT. IF THE SIZE BIT (BIT 7-\$ENVM) IS CLEAR THE PROGRAM WILL OPERATE IN NORMAL STAND-ALONE MODE. IF THE SIZE BIT IS SET THE PROGRAM WILL COMPARE PARAMETERS FROM THE BOARD TO THE CONTENTS OF THE ETABLE. TO USE THE APT COMPARSION FEATURE THE ETABLE MUST BE SET UP IN THIS ORDER;

\$ENV:	DON'T CARE
\$ENVM:	200 OR 240
\$SWREG:	DON'T CARE
\$USWR:	NOT USED
\$CPUOP:	NOT USED
\$MAMS1:	NOT USED
\$MTYP1:	NOT USED
\$MADR1:	NOT USED
\$MAMS2:	NOT USED
\$MTYP2:	NOT USED
\$MADR2:	NOT USED
\$MAMS3:	NOT USED
\$MTYP3:	NOT USED
\$MADR3:	NOT USED
\$MAMS4:	NOT USED
\$MAMS4:	NOT USED
\$MTYP4:	NOT USED
\$MADR4:	NOT USED
\$VECT1:	NOT USED
\$VECT2:	NOT USED
\$BASE:	PSEUDO POWER-FAIL VECTOR ADDRESS
\$DEVN:	NOT USED
\$CDW1:	CONTENTS OF ADDRESS IN \$BASE
\$CDW2:	NOT USED
\$DDW0:	DEVICE CODES EXPECTED
	FIRST LETTER/SECOND LETTER
DDW15:	

5.0 SUBROUTINE ABSTRACTS

5.1 NOROMS

THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND DURING SIZING IT DOES A READ OF ALL BOOTSTRAP ROM ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN EXPECTED VALUE.

5.2 CHECKS

THIS ROUTINE SETS UP THE FIRST, LAST AND EXCEPTION ADDRESSES FOR THE "CALSUM" SUBROUTINE. IT RECEIVES THE CALCULATED CHECKSUM FROM "CALSUM" AND COMPARES IT AGAINST THE GOOD CHECKSUM TO DETERMINE CRC ERRORS.

5.3 CALSUM

THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF EACH ROM. IT RECEIVES THE FIRST ADDRESS TO BE CHECKED(FIRSTA), THE LAST ADDRESS TO BE CHECKED(LASTAD) AND THE EXCEPTION ADDRESS (EXCADD) FROM THE "CHECKS" MODULE AND RETURNS TO IT THE CHECKSUM IN R4.

5.4 PROMP

THIS ROUTINE PROCESSES THE ROM PARAMETERS. IT CHECKS THE SIZE/DON'T SIZE BIT IN THE APT ETABLE AND EITHER FORMATS THE SIZING MESSAGES OR COMPARES THE SIZING INFORMATION TO THE APT ETABLE DATA.

5.5 DEVCOD

THIS ROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE ADDRESS IN WHICH IT WAS FOUND BACK TO THE "PROMP" MODULE.

5.6 PPFVAR

THIS ROUTINE DETERMINES THE PSEUDO POWER-FAIL VECTOR ADDRESS AND PASSES IT AND ITS CONTENTS TO THE "PROMP" MODULE

5.7 PUTMES

THIS ROUTINE FORMATS THE PARAMETER AND POWER-FAIL ADDRESS MESSAGES.

5.8 ERRHAN

THIS ROUTINE FORMATS ALL ERROR MESSAGES AND CALLS THE TYPE ROUTINE TO OUTPUT THEM. IT ALSO USES THE OPERATIONAL SWITCHES TO DETERMINE WHETHER TO OUTPUT THE MESSAGE, OR HALT.

5.9 FILBUF

THIS ROUTINE FILLS THE MESSAGE BUFFER WITH ASCII CHARACTERS. IT RECIEVES THE ADDRESS OF THE ASCII IN R5.

5.10 OCASC

THIS ROUTINE TAKES A SIXTEEN BIT BINARY NUMBER AND CONVERTS IT TO 6 ASCII CHARACTERS.

5.11 OCADD

THIS ROUTINE IS USED BY THE "PUTMES" MODULE WHEN THE DATA IN R3 IS NOT IN THE RIGHT MODE TO BE HANDLED BY THE "OCASC" MODULE. IT MOVES THE ADDRESSING MODE OF R3 UP ONE LEVEL OF DEFERMENT.

6.0 RELIABILITY//AVAILABILITY/SERVICEABILITY

WHEN RUNNING IN ANY ENVIRONMENT BUT APT THERE IS ONLY ONE ERROR DETECTED BY THE PROGRAM. THIS ERROR IS A CHECKSUM ERROR. THE MESSAGE WILL BE:

CRC ERROR IN ROM EXX

WHEN RUNNING IN THE APT ENVIRONMENT THREE OTHER ERRORS MAY OCCUR.

1. AN ERROR WILL OCCUR WHEN THE DEVICE CODES IN THE ETABLE DO NOT MATCH THE DEVICE CODES FOUND IN THE ROMS. THE ERROR MESSAGE WILL BE EITHER:

1. COULD NOT FIND DEVICE CODE XX.
2. FOUND UNEXPECTED DEVICE CODE XX.

THE FIRST MESSAGE WILL BE PASSED TO APT IF A DEVICE CODE LISTED IN THE ETABLE CANNOT BE FOUND IN THE EXISTING ROMS. THE SECOND MESSAGE WILL BE SENT IF A DEVICE CODE NOT LISTED IN THE ETABLE IS FOUND IN A ROM.

2. THE SECOND ERROR WILL BE IF THE PSEUDO POWER-FAIL VECTOR ADDRESS IN THE ETABLE DOES NOT MATCH THE ADDRESS DETERMINED BY THE PROGRAM. TO BE IN THE BOARDS' SWITCHES. THE MESSAGE WILL BE:

POWER-FAIL VECTOR
EXPECTED RECEIVED

WHERE EXPECTED IS THE CONTENTS OF THE ETABLE AND RECEIVED IS THE VALUE FOUND BY THE PROGRAM.

3. THE THIRD ERROR WILL BE IF THE CONTENTS OF BOARD'S PSEUDO POWER-FAIL VECTOR ADDRESS DOES NOT MATCH THE VALUE EXPECTED FROM THE ETABLE. THE MESSAGE WILL BE:

POWER-FAIL DATA ERROR
EXPECTED RECEIVED

!

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099
000100
000101
000102
000103
000104
000105
000106
000107
000108
000109
000110
000111
000112
000113
000114
000115
000116
000117
000118
000119
000120
000121
000122
000123
000124
000125
000126
000127
000128
000129
000130
000131
000132
000133
000134
000135
000136
000137
000138
000139
000140
000141
000142
000143
000144
000145
000146
000147
000148
000149
000150
000151
000152
000153
000154
000155
000156
000157
000158
000159
000160
000161
000162
000163
000164
000165
000166
000167
000168
000169
000170
000171
000172
000173
000174
000175
000176
000177
000178
000179
000180
000181
000182
000183
000184
000185
000186
000187
000188
000189
000190
000191
000192
000193
000194
000195
000196
000197
000198
000199
000200
000201
000202
000203
000204
000205
000206
000207
000208
000209
000210
000211
000212
000213
000214
000215
000216
000217
000218
000219
000220
000221
000222
000223
000224
000225
000226
000227
000228
000229
000230
000231
000232
000233
000234
000235
000236
000237
000238
000239
000240
000241
000242
000243
000244
000245
000246
000247
000248
000249
000250
000251
000252
000253
000254
000255
000256
000257
000258
000259
000260
000261
000262
000263
000264
000265
000266
000267
000268
000269
000270
000271
000272
000273
000274
000275
000276
000277
000278
000279
000280
000281
000282
000283
000284
000285
000286
000287
000288
000289
000290
000291
000292
000293
000294
000295
000296
000297
000298
000299
000300
000301
000302
000303
000304
000305
000306
000307
000308
000309
000310
000311
000312
000313
000314
000315
000316
000317
000318
000319
000320
000321
000322
000323
000324
000325
000326
000327
000328
000329
000330
000331
000332
000333
000334
000335
000336
000337
000338
000339
000340
000341
000342
000343
000344
000345
000346
000347
000348
000349
000350
000351
000352
000353
000354
000355
000356
000357
000358
000359
000360
000361
000362
000363
000364
000365
000366
000367
000368
000369
000370
000371
000372
000373
000374
000375
000376
000377
000378
000379
000380
000381
000382
000383
000384
000385
000386
000387
000388
000389
000390
000391
000392
000393
000394
000395
000396
000397
000398
000399
000400
000401
000402
000403
000404
000405
000406
000407
000408
000409
000410
000411
000412
000413
000414
000415
000416
000417
000418
000419
000420
000421
000422
000423
000424
000425
000426
000427
000428
000429
000430
000431
000432
000433
000434
000435
000436
000437
000438
000439
000440
000441
000442
000443
000444
000445
000446
000447
000448
000449
000450
000451
000452
000453
000454
000455
000456
000457
000458
000459
000460
000461
000462
000463
000464
000465
000466
000467
000468
000469
000470
000471
000472
000473
000474
000475
000476
000477
000478
000479
000480
000481
000482
000483
000484
000485
000486
000487
000488
000489
000490
000491
000492
000493
000494
000495
000496
000497
000498
000499
000500
000501
000502
000503
000504
000505
000506
000507
000508
000509
000510
000511
000512
000513
000514
000515
000516
000517
000518
000519
000520
000521
000522
000523
000524
000525
000526
000527
000528
000529
000530
000531
000532
000533
000534
000535
000536
000537
000538
000539
000540
000541
000542
000543
000544
000545
000546
000547
000548
000549
000550
000551
000552
000553
000554
000555
000556
000557
000558
000559
000560
000561
000562
000563
000564
000565
000566
000567
000568
000569
000570
000571
000572
000573
000574
000575
000576
000577
000578
000579
000580
000581
000582
000583
000584
000585
000586
000587
000588
000589
000590
000591
000592
000593
000594
000595
000596
000597
000598
000599
000600
000601
000602
000603
000604
000605
000606
000607
000608
000609
000610
000611
000612
000613
000614
000615
000616
000617
000618
000619
000620
000621
000622
000623
000624
000625
000626
000627
000628
000629
000630
000631
000632
000633
000634
000635
000636
000637
000638
000639
000640
000641
000642
000643
000644
000645
000646
000647
000648
000649
000650
000651
000652
000653
000654
000655
000656
000657
000658
000659
000660
000661
000662
000663
000664
000665
000666
000667
000668
000669
000670
000671
000672
000673
000674
000675
000676
000677
000678
000679
000680
000681
000682
000683
000684
000685
000686
000687
000688
000689
000690
000691
000692
000693
000694
000695
000696
000697
000698
000699
000700
000701
000702
000703
000704
000705
000706
000707
000708
000709
000710
000711
000712
000713
000714
000715
000716
000717
000718
000719
000720
000721
000722
000723
000724
000725
000726
000727
000728
000729
000730
000731
000732
000733
000734
000735
000736
000737
000738
000739
000740
000741
000742
000743
000744
000745
000746
000747
000748
000749
000750
000751
000752
000753
000754
000755
000756
000757
000758
000759
000760
000761
000762
000763
000764
000765
000766
000767
000768
000769
000770
000771
000772
000773
000774
000775
000776
000777
000778
000779
000780
000781
000782
000783
000784
000785
000786
000787
000788
000789
000790
000791
000792
000793
000794
000795
000796
000797
000798
000799
000800
000801
000802
000803
000804
000805
000806
000807
000808
000809
000810
000811
000812
000813
000814
000815
000816
000817
000818
000819
000820
000821
000822
000823
000824
000825
000826
000827
000828
000829
000830
000831
000832
000833
000834
000835
000836
000837
000838
000839
000840
000841
000842
000843
000844
000845
000846
000847
000848
000849
000850
000851
000852
000853
000854
000855
000856
000857
000858
000859
000860
000861
000862
000863
000864
000865
000866
000867
000868
000869
000870
000871
000872
000873
000874
000875
000876
000877
000878
000879
000880
000881
000882
000883
000884
000885
000886
000887
000888
000889
000890
000891
000892
000893
000894
000895
000896
000897
000898
000899
000900
000901
000902
000903
000904
000905
000906
000907
000908
000909
000910
000911
000912
000913
000914
000915
000916
000917
000918
000919
000920
000921
000922
000923
000924
000925
000926
000927
000928
000929
000930
000931
000932
000933
000934
000935
000936
000937
000938
000939
000940
000941
000942
000943
000944
000945
000946
000947
000948
000949
000950
000951
000952
000953
000954
000955
000956
000957
000958
000959
000960
000961
000962
000963
000964
000965
000966
000967
000968
000969
000970
000971
000972
000973
000974
000975
000976
000977
000978
000979
000980
000981
000982
000983
000984
000985
000986
000987
000988
000989
000990
000991
000992
000993
000994
000995
000996
000997
000998
000999
001000

```

.ENABLE ABS
.LIST ME
.NLIST MC,MD,CND
.TITLE CZM9BAD M9312 BOOT TERMR BK
.*COPYRIGHT (C) 1978
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY BARRY G. IRRGANG
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
$TN=1
$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----
.* 15 HALT ON ERROR
.* 14 LOOP ON TEST
.* 13 INHIBIT ERROR TYPEOUTS
.SBTTL BASIC DEFINITIONS

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

.*MISCELLANEOUS DEFINITIONS
MT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0 ;;GENERAL REGISTER
R1= %1 ;;GENERAL REGISTER
R2= %2 ;;GENERAL REGISTER
R3= %3 ;;GENERAL REGISTER
R4= %4 ;;GENERAL REGISTER
R5= %5 ;;GENERAL REGISTER
R6= %6 ;;GENERAL REGISTER
R7= %7 ;;GENERAL REGISTER
SP= %6 ;;STACK POINTER
PC= %7 ;;PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS
PRO= 0 ;;PRIORITY LEVEL 0

```

000001
160000

001100

000011
000012
000015
000200
177776
177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

000000

I01

CZM98A0 M9312 BOOT TERMR 8K
CZM98A.P11 13-MAR-78 08:58

MACY11 30A(1052) 13-MAR-78 09:59 PAGE 8
BASIC DEFINITIONS

SEG 0008

354 000040
355 000100
356 000140
357 000200
358 000240
359 000300
360 000340
361
362
363 100000
364 040000
365 020000
366 010000
367 004000
368 002000
369 001000
370 000400
371 000200
372 000100
373 000040
374 000020
375 000010
376 000004
377 000002
378 000001
379
380
381
382
383
384
385
386
387
388
389
390
391 100000
392 040000
393 020000
394 010000
395 004000
396 002000
397 001000
398 000400
399 000200
400 000100
401 000040
402 000020
403 000010
404 000004
405 000002
406 000001
407
408
409

PR1= 40
PR2= 100
PR3= 140
PR4= 200
PR5= 240
PR6= 300
PR7= 340
::PRIORITY LEVEL 1
::PRIORITY LEVEL 2
::PRIORITY LEVEL 3
::PRIORITY LEVEL 4
::PRIORITY LEVEL 5
::PRIORITY LEVEL 6
::PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
.EQUIV SW09,SW9
.EQUIV SW08,SW8
.EQUIV SW07,SW7
.EQUIV SW06,SW6
.EQUIV SW05,SW5
.EQUIV SW04,SW4
.EQUIV SW03,SW3
.EQUIV SW02,SW2
.EQUIV SW01,SW1
.EQUIV SW00,SW0

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7

BASIC DEFINITIONS

```

410 .EQUIV BIT06,BIT6
411 .EQUIV BIT05,BIT5
412 .EQUIV BIT04,BIT4
413 .EQUIV BIT03,BIT3
414 .EQUIV BIT02,BIT2
415 .EQUIV BIT01,BIT1
416 .EQUIV BIT00,BIT0
417
418
419 000004
420 000010
421 000014
422 000014
423 000014
424 000020
425 000024
426 000030
427 000034
428 000060
429 000064
430 000240
431
432
433
434 000001
435 000002
436 000004
437 000010
438 000020
439 000040
440 000100
441 000200
442
443 000000
444
445
446 000000
447
448
449
450
451 000174 000000
452 000176 000000
453
454 000200 000137 001400
455 000204 000204
456 000204 000167 001634
457
458
459
460
461
462 000210
463 000046
464 000046 002152
465 000052

```

```

.*BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
RESVEC= 10        ;; RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14        ;; "T" BIT
TRTVEC= 14        ;; TRACE TRAP
BPTVEC= 14        ;; BREAKPOINT TRAP (BPT)
IOTVEC= 20        ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24        ;; POWER FAIL
EMTVEC= 30        ;; EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34        ;; "TRAP" TRAP
TKVEC= 60         ;; TTY KEYBOARD VECTOR
TPVEC= 64         ;; TTY PRINTER VECTOR
PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
;*****
;* PROGRAM EQUATES
;*****
APTER1 = 1
APTER2 = 2
APTER3 = 4
APTER4 = 10
CRCERR = 20
PFERR = 40
NOROME =100
SEQERR =200

.=0
.SBTTL TRAP CATCHER

.=0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0       ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
.=204
JMP RSTART

.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
$SVPC=.           ;SAVE PC
.=46
$ENDAD           ;; 1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
.=52

```

```

466 000052 000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
467          000210          .=SSVPC        ;; RESTORE PC
468
469          001100          .=1100
470 001100          000      $AUTOB: .BYTE 0
471 001101          000      $INTAG: .BYTE 0
472 001102 000000          .WORD 0
473 001104 177570          SWR: .WORD DSWR
474 001106 177570          DISPLAY: .WORD DDISP
475 001110 000000          ROMERR: 0
476 001112 000000          MESSAG: 0
477 001114 173000          FIRSTB: 173000
478 001116 000000          ROMFIN: 0
479 001120 000000          ERRCNT: 0
480
481          .SBTTL APT PARAMETER BLOCK
482
483          ;*****
484          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
485          ;*****
486          001122          .SX=.          ;;SAVE CURRENT LOCATION
487          000024          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
488 000024 000200          200          ;;FOR APT START UP
489          000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
490 000044 001122          $APTHDR ;;POINT TO APT HEADER BLOCK
491          001122          .=.SX        ;;RESET LOCATION COUNTER
492          ;*****
493          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
494          ;INTERFACE SPEC.
495
496 001122          $APTHD:
497 001122 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
498 001124 001136          $MBADR: .WORD $MAIL        ;;ADDRESS OF APT MAILBOX (BITS 0-15)
499 001126 000002          $STMT: .WORD 2.          ;;RUN TIM OF LONGEST TEST
500 001130 000002          $PASTM: .WORD 2.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
501 001132 000000          $UNITM: .WORD 0          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
502 001134 000052          .WORD SETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
503          .SBTTL APT MAILBOX-ETABLE
504
505          ;*****
506          .EVEN
507 001136          $MAIL:          ;;APT MAILBOX
508 001136 000000          $MSGTY: .WORD AMSGTY        ;;MESSAGE TYPE CODE
509 001140 000000          $FATAL: .WORD AFATAL        ;;FATAL ERROR NUMBER
510 001142 000000          $TESTN: .WORD ATESTN        ;;TEST NUMBER
511 001144 000000          $PASS: .WORD APASS          ;;PASS COUNT
512 001146 000000          $DEVCT: .WORD ADEVCT        ;;DEVICE COUNT
513 001150 000000          $UNIT: .WORD AUNIT        ;;I/O UNIT NUMBER
514 001152 000000          $MSGAD: .WORD AMSGAD        ;;MESSAGE ADDRESS
515 001154 000000          $MSGLG: .WORD AMSGLG        ;;MESSAGE LENGTH
516 001156          $ETABLE:          ;;APT ENVIRONMENT TABLE
517 001156          $ENV: .BYTE AENV          ;;ENVIRONMENT BYTE
518 001157          $ENVM: .BYTE AENVM        ;;ENVIRONMENT MODE BITS
519 001160 000000          $SWREG: .WORD ASWREG        ;;APT SWITCH REGISTER
520 001162 000000          $USWR: .WORD AUSWR          ;;USER SWITCHES
521 001164 000000          $CPUOP: .WORD ACPUOP        ;;CPU TYPE,OPTIONS

```


MO1

CZM9BA0 M9312 BOOT TERMR BK MACY11 30A(1052) 13-MAR-78 09:59 PAGE 12
 CZM9BA.P11 13-MAR-78 08:58 INITIALIZE THE COMMON TAGS

SEQ 0012

```

578 001420 005067 177520          CLR    $PASS          ;; CLEAR THE PASS COUNT
579 001424 016767 000476 000466  MOV    SENDCT,SEOPCT  ;; SETUP END-OF-PROGRAM COUNTER
580                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
581                                     ;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
582 001432 013746 000004          MOV    @ERRVEC, -(SP)  ;; SAVE ERROR VECTOR
583 001436 012737 001472 000004  MOV    @64$, @ERRVEC  ;; SET UP ERROR VECTOR
584 001444 012767 177570 177432  MOV    @DSWA, SWR     ;; SETUP FOR A HARDWARE SWICH REGISTER
585 001452 012767 177570 177426  MOV    @DDISP, DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
586 001460 022777 177777 177416  CMP    #-1, @SWR     ;; TRY TO REFERENCE HARDWARE SWR
587 001466 001012          BNE    66$           ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
588                                     ;; AND THE HARDWARE SWR IS NOT = -1
589 001470 000403          BR     65$           ;; BRANCH IF NO TIMEOUT
590 001472 012716 001500 64$:    MOV    @65$, (SP)    ;; SET UP FOR TRAP RETURN
591 001476 000002          RTI
592 001500 012767 000176 177376 65$:    MOV    @SWREG, SWR   ;; POINT TO SOFTWARE SWR
593 001506 012767 000174 177372  MOV    @DISPREG, DISPLAY
594 001514 012637 000004 66$:    MOV    (SP)+, @ERRVEC ;; RESTORE ERROR VECTOR
595
596 001520 005067 177420          CLR    $PASS          ;; CLEAR PASS COUNT
597 001524 132767 000200 177425  BITB   @APTSIZE, $ENVM ;; TEST USER SIZE UNDER APT
598 001532 001403          BEQ    67$           ;; YES, USE NON-APT SWITCH
599 001534 012767 001160 177342  MOV    @SSWREG, SWR  ;; NO, USE APT SWITCH REGISTER
600 001542
601                                     .SBTTL  TYPE PROGRAM NAME
602                                     ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
603 001542 005227 177777          INC    #-1           ;; FIRST TIME?
604 001546 001046          BNE    68$           ;; BRANCH IF NO
605 001550 022737 002152 000042  CMP    @SENDAD, @#42 ;; ACT-11?
606 001556 001442          BEQ    68$           ;; BRANCH IF YES
607 001560 104401 001626          TYPE   69$           ;; TYPE ASCIZ STRING
608                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
609 001564 005737 000042          TST    @#42          ;; ARE WE RUNNING UNDER XXDP/ACT?
610 001570 001012          BNE    70$           ;; BRANCH IF YES
611 001572 126727 177360 000001  CMPB   $ENV, #1      ;; ARE WE RUNNING UNDER APT?
612 001600 001406          BEQ    70$           ;; BRANCH IF YES
613 001602 026727 177276 000176  CMP    SWR, @SWREG   ;; SOFTWARE SWITCH REG SELECTED?
614 001610 001005          BNE    71$           ;; BRANCH IF NO
615 001612 104405          GTSWR  ;; GET SOFT-SWR SETTINGS
616 001614 000403          BR     71$
617 001616 112767 000001 177254 70$:    MOV    @1, $AUTOB   ;; SET AUTO-MODE INDICATOR
618 001624 71$:
619 001624 000417          BR     68$           ;; GET OVER THE ASCIZ
620                                     ;; 69$: .ASCIZ <CRLF>*CZM9BA0 M9312 BOOT TERMR BK*<CRLF>
621 001664 68$:
622 001664 012700 007420          MOV    @BUF1, RO    ;; CLR SIZING BUFFERS
623 001670 005020          CLR    (RO)+
624 001672 020027 007504 8$:    CMP    RO, @OCTBUF  ;; HAVE WE CLEARED THE WHOLE
625 001676 002774          BLT    8$           ;; BUFFER, NO, GO BACK
626 001700 005037 001116          CLR    @ROMFIN     ;; INITIALIZE ROM FOUND INDICATORS
627 001704 005037 003644          CLR    @TIMES      ;; INITIALIZE ENTRY COUNTER FOR PUTMES SUB
628 001710 013746 000004          MOV    @ERRVEC, -(R6) ;; SAVE CONTENTS OF LOCATION 4
629 001714 016737 000036 000004  MOV    @ERRVEC, @#165000 ;; SET UP FOR POSSIBLE TRAP
630 001722 122737 177777 165000  CMP    #-1, @#165000 ;; IF CONTENTS = -1, OR TRAP
631 001730 001414          BEQ    3$           ;; THEN CPU ROM NOT PLUGGED IN
632 001732 012700 165000          MOV    @165000, RO  ;; GET FIRST ADDRESS OF ROM SPACE
633 001736 005720 1$:    TST    (RO)+       ;; IF THIS INSTRUCTION TRAPS CPU ROM
    
```


690 002160 000240
691 002162
692 002162 000137
693 002164 002044
694 002166 377 000
695 002171 015 042412 042116
696 002176 050040 051501 000123

NOP ;:ACT11
\$DOAGN: ;:RETURN
\$RTNAD: .WORD @PC)+
\$ENULL: .BYTE -1,-1 0 ;:NULL CHARACTER STRING
\$ENDMG: .ASCIZ <15><12>/END PASS/

697
698
699

700
701
702
703
704
705

;NO ROMS TEST
;THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND
;DURING SIZING. IT DOES A READ OF ALL BOOTSTRAP ROM
;ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN
;EXPECTED VALUE.

706 002204 012703 173000
707 002210 013704 002242
708 002214 022304
709 002216 001405
710 002220 052737 000100 001110
711 002226 004767 001630
712 002232 022703 174000
713 002236 003366
714 002240 000207

NOROMS: MOV #173000, R3 ;GET FIRST ADDRESS TO BE READ
MOV @NODATA, R4 ;GET ADDRESS OF EXPECTED VALUE
1\$: CMP (R3)+, R4 ;DOES RECIEVED VALUE EQUAL EXPECTED
BEQ 2\$, ;IF YES CONTINUE TESTING
BIS #NOROME, @#ROMERR ;IF NO SET ERROR INDICATER
JSR PC, ERRHAN ;REPORT ERROR
2\$: CMP #174000, R3 ;HAVE ALL ADDRESSES BEEN TESTED
BGT 1\$, ;IF NO GO TEST THIS ADDRESS
RTS PC

715
716 002242 161777

NODATA: .WORD 161777

717
718
719
720

721
722
723

;CHECKSUM ROMS MODULE
;THIS ROUTINE DOES A CHECKSUM OF ALL ROMS PRESENT

724 002244 012737 000001 002460
725 002252 033737 002460 001116
726 002260 001424
727 002262 012737 165775 002556
728 002270 012737 000000 002554
729 002276 012737 165000 002552
730 002304 004767 000152
731 002310 013703 165776
732 002314 020304
733 002316 001405
734 002320 052737 000020 001110
735 002326 004767 001530
736 002332 012737 173000 002552
737 002340 012737 173024 002554
738 002346 012737 173175 002556
739 002354 012702 173176
740 002360 006137 002460
741 002364 033737 002460 001116
742 002372 001412
743 002374 004767 000062
744 002400 011203
745 002402 020304

CHECKS: MOV #1, @#ROMCNT ;INITIALIZE ROM COUNTER
BIT @#ROMCNT, @#ROMFIN ;IS DIAGNOSTIC ROM PRESENT
1\$: BEQ 1\$, ;IF NO GO CHECKSUM BOOT ROMS
MOV #165775, @#LASTAD ;SET UP LAST ADDRESS TO BE SUMMED
MOV #0, @#EXCADD ;SET UP EXCEPTION ADDRESS
MOV #165000, @#FIRSTA ;SET UP FIRST ADDRESS TO BE SUMMED
JSR PC, CALSUM ;GO CALCULATE CHECKSUM
MOV @#165775, R3 ;GET EXPECTED CHECKSUM
CMP R3, R4 ;COMPARE CHECKSUMS
BEQ 1\$, ;IF CHECKSUMS COMPARE CONTINUE
BIS #CRCERR, @#ROMERR ;IF ERROR SET INDICATER
JSR PC, ERRHAN ;REPORT ERROR
2\$: MOV #173000, @#FIRSTA ;SET UP FIRST ADDRESS TO BE SUMMED
MOV #173024, @#EXCADD ;SET UP EXCEPTION ADDRESS
MOV #173175, @#LASTAD ;SET UP LAST ADDRESS TO BE SUMMED
MOV #173176, R2 ;GET ADDRESS OF GOOD DATA
3\$: ROL @#ROMCNT, @#ROMFIN ;UPDATE ROM COUNTER
BIT @#ROMCNT, @#ROMFIN ;IS ROM PRESENT
BEQ 3\$, ;IF NO, GO UPDATE TO NEXT ROM
JSR PC, CALSUM ;IF YES GO CALCULATE CHECKSUM
MOV (R2), R3 ;GET EXPECTED CHECKSUM
CMP R3, R4 ;COMPARE CHECKSUMS

```

746 002404 001405          BEQ      3$
747 002406 052737 000020 001110      BIS      #CRCERR, @#ROMERR ; IF CHECKSUMS EQUAL CONTINUE
748 002414 004767 001442          JSR      PC          ERRHAN ; IF ERROR SET INDICATER
749 002420 062737 000200 002552 3$:      ADD      #200, @#FIRSTA ; REPORT ERROR
750 002426 062737 000200 002554      ADD      #200, @#EXCADD ; UPDATE FIRST ADDRESS
751 002434 062737 000200 002556      ADD      #200, @#LASTAD ; UPDATE EXCEPTION ADDRESS
752 002442 062702 000200          ADD      #200, R2 ; UPDATE LAST ADDRESS
753 002446 022737 174000 002552      CMP      #174000, @#FIRSTA ; UPDATE ADDRESS OF GOOD DATA
754 002454 001341          BNE      2$          ; HAVE ALL ROMS BEEN CHECKED
755 002456 000207          RTS      PC          ; IF NO GO CHECKSUM NEXT ONE
756                                     ; IF YES RETURN

```

757 002460 000000 ROMCNT: 0

```

758
759
760 ;CALCULATE CHECKSUMS MODULE
761 ;THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTEI..S
762 ;OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS
763 ;OF EVERY ROM
764
765

```

```

766 002462 010246          CALSUM: MOV      R2          -(SP) ; SAVE R2
767 002464 013700 002552      MOV      @#FIRSTA, R0 ; GET STARTING ADDRESS
768 002470 005004          CLR      R4 ; INITIALIZE CRC WORD
769 002472 012005          LOOP: MOV      (R0)+, R5 ; GET A BYTE
770 002474 012702 000020      MOV      #16., R2 ; SET BYTE COUNT
771 002500 000241          CRCLOP: CLC ; THE NEXT NINE LINES
772 002502 006004          ROR      R4 ; DO THE MATH CALCULATIONS
773 002504 006005          ROR      R5
774 002506 102006          BVC      1$
775 002510 012701 120001      MOV      #120001, R1
776 002514 040401          BIC      R4, R1
777 002516 042704 120001      BIC      #120001, R4
778 002522 050104          BIS      R1, R4
779 002524 005302          1$:      DEC      R2
780 002526 003364          BGT      CRCLOP
781 002530 020037 002554      CMP      R0, @#EXCADD ; IS NEXT ADDRESS AN EXCEPTION ADDRESS
782 002534 001001          BNE      2$          ; IF NO TEST IT
783 002536 005720          TST      (R0)+ ; IF YES SKIP ADDRESS
784 002540 020037 002556      2$:      CMP      R0, @#LASTAD ; HAVE ALL LOCATIONS BEEN SUMMED
785 002544 101752          BLOS     LOOP ; IF NO CONTINUE
786 002546 012602          MOV      (SP)+, R2 ; RESTORE R2
787 002550 000207          RTS      PC          ; IF YES RETURN

```

```

788
789 002552 000000          FIRSTA: 0
790 002554 000000          EXCADD: 0
791 002556 000000          LASTAD: 0
792
793
794

```

```

795 ;PROCESS ROM PARAMETERS MODULE
796 ;THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE. IF IT IS THE
797 ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR
798 ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT. IF SIZING IS
799 ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED
800 ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE.
801

```


E02

CZM9BA0 M9312 BOOT TERM 8K
CZM9BA.P11 13-MAR-78 08:58

MACY11 30A(1052) 13-MAR-78 09:59 PAGE 17
END OF PASS ROUTINE

SEG 0017

```

858 003042 004767 001014          JSR   PC,          ERRHAN          ;GO TO ERROR ROUTINE
859 003046 000207          9$:   RTS   PC
860
861          ;GET DEVICE CODES MODULE
862          ;THIS SUBROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
863          ;ADDRESS IN WHICH IT WAS FOUND BACK TO THE CALLING ROUTINE.
864          ;DATA IS STORED IN BUFFER "BUF1" IN THIS FORMAT:
865          ;
866          ;           BUF1:  ADDRESS OF FIRST DEVICE CODE
867          ;                   DEVICE CODE
868          ;                   ADDRESS OF SECOND DEVICE CODE
869          ;                   DEVICE CODE
870          ;                   .
871          ;                   .
872          ;                   .
873          ;                   .
874          ;                   ADDRESS OF NTH DEVICE CODE
875          ;                   DEVICE CODE
876          ;
877          ;IF DIAGNOSTIC ROM PRESENT IT WILL BE THE FIRST DEVICE CODE.
878          ;THERE IS ROOM FOR UP TO 12 DEVICE CODES IN TEMP
879 003050 012700 007420          DEVCOD: MOV   #BUF1,          R0
880 003054 012701 000001          MOV   #1,             R1
881 003060 030137 001116          BIT   R1,             @#ROMFIN          ; INITIALIZE ROM POINTER
882 003064 001411          BEQ   1$,             @#ROMFIN          ; IS DIAGNOSTIC ROM PRESENT
883 003066 012737 165774 003250          MOV   #165774,        @#TESTAD          ; IF NO GO TEST BOOT ROMS
884 003074 013720 003250          MOV   @#TESTAD,        (R0)+          ; GET ADDRESS OF DIAG. ROM DEVICE CODE.
885 003100 017720 000144          MOV   @TESTAD,        (R0)+          ; STORE ADDRESS OF DEVICE CODE
886 003104 005037 003252          CLR   @#DAFLAG          ; STORE DEVICE CODE
887 003110 013737 001114 003250 1$:   MOV   @#FIRSTB,        @#TESTAD          ; CLR DATA TYPE FLAG
888 003116 006101          ROL   R1
889 003120 030137 001116          3$:   BIT   R1,             @#ROMFIN          ; GET ADDRESS OF FIRST BOOT ROM
890 003124 001005          BNE   4$,             @#TESTAD          ; UPDATE POINTER TO NEXT ROM
891 003126 062737 000200 003250          ADD   #200,           @#TESTAD          ; IS ROM PRESENT
892 003134 006101          ROL   R1
893 003136 000437          BR   9$
894 003140 005737 003252          4$:   TST   @#DAFLAG          ; IF YES GO GET PARAMETERS
895 003144 001010          BNE   5$,             @#TESTAD          ; IF NO UPDATE TEST ADDRESS
896 003146 013720 003250          MOV   @#TESTAD,        (R0)+          ; UPDATE POINTER TO NEXT ROM
897 003152 017720 000072          MOV   @TESTAD,        (R0)+          ; GO SEE IF ALL ROM CHECKED
898 003156 062737 000002 003250          ADD   #2,             @#TESTAD          ; IF DATAFLAG=0 THEN DATA IS DEVICE CODE
899 003164 000422          BR   8$
900 003166 013702 003250          5$:   MOV   @#TESTAD,        R2
901 003172 067737 000052 003250          ADD   @TESTAD,        @#TESTAD          ; STORE ADDRESS OF DEVICE CODE
902 003200 013703 003250          MOV   @TESTAD,        R3
903 003204 042702 177177          BIC   #177177,        R2
904 003210 042703 177177          BIC   #177177,        R3
905 003214 160203          SUB   R2,             R3
906 003216 005703          6$:   TST   R3
907 003220 001404          BEQ   8$
908 003222 162703 000200          SUB   #200,           R3
909 003226 006101          ROL   R1
910 003230 000772          BR   6$
911 003232 005137 003252          8$:   COM   @#DAFLAG          ; CHANGE DATA FLAG TO RIGHT DATA TYPE
912 003236 023727 003250 174000 9$:   CMP   @#TESTAD,        #174000          ; HAVE WE CHECKED ALL THE ROM
913 003244 002725          BLT   3$
          ; IF NO CONTINUE

```

```

914 003246 000207          RTS      PC          ; IF YES RETURN
915
916 003250 000000          TESTAD: 0
917 003252 000000          DAFLAG: 0
918
919          ; GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
920          ; THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
921          ; WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
922          ; BOARD. THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
923          ; SWITCH MUST BE "ON" OR AN ERROR WILL BE DETECTED.
924          ; THE DATA WILL BE RETURNED IN "BUF2" IN THE FORMAT.
925          ;
926          ;          BUF2:  PSEUDO POWER-FAIL VECTOR ADDRESS
927          ;          ;          CONTENTS OF VECTOR ADDRESS
928          ;
929 003254 032737 000777 173024 PPFVAR: BIT      #777,          @#173024      ; TEST IF LOCATION 173024 SELECTED
930 003262 001407          BEQ      1$          ; IF NOT THEN GO TEST LOCATION 173224
931 003264 012737 173024 007500      MOV      #173024,     @#BUF2          ; IF IT IS THEN STORE ADDRESS
932 003272 013737 173024 007502      MOV      @#173024,   @#BUF2+2        ; STORE CONTENTS OF LOCATION 173024
933 003300 000416          BR       3$          ; GO TO RETURN
934 003302 032737 000777 173224 1$: BIT      #777,          @#173224      ; TEST IF LOCATION 173224 SELECTED
935 003310 001407          BEQ      2$          ; IF NOT THEN SET ERROR INDICATOR
936 003312 012737 173224 007500      MOV      #173224,     @#BUF2          ; IF IT IS THEN STORE ADDRESS
937 003320 013737 173224 007502      MOV      @#173224,   @#BUF2+2        ; STORE CONTENTS OF VECTOR
938 003326 000403          BR       3$          ; GET OVER ERROR
939 003330 052737 000040 001110 2$: BIS      #PFERR,     @#ROMERR        ; SET ERROR INDICATOR
940 003336 000207          3$:      RTS      PC
941
942
943
944
945          ; PUT MESSAGE IN BUFFER ROUTINE
946          ; THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.
947
948 003340 017604 000000          PUTMES: MOV      @ (R6),          R4          ; GET MESSAGE BUFFER ADDRESS
949 003344 005737 003644          TST      @#TIMES          ; IS THIS FIRST TIME THROUGH
950 003350 001076          BNE      3$          ; IF NOT FIRST TIME THEN FORMAT POWER-
951          ; FAIL MESSAGE
952          ; TOGGLE WATCHDOG
953 003352 005237 003644          INC      @#TIMES          ; GET DATA BUFFER ADDRESS
954 003356 012703 007420          MOV      #BUF1,          R3          ; IS DIAGNOSTIC ROM PRESENT
955 003362 022713 165774          CMP      #165774,        (R3)          ; IF NOT DON'T FORMAT DIAG. ROM MESSAGE
956 003366 001012          BNE      1$          ; IF IT IS GET DIAG. ROM MESSAGE HEADER
957 003370 012705 006530          MOV      #DRHEAD,       R5          ; GO PUT HEADER IN MESSAGE BUFFER
958 003374 004767 001000          JSR      PC,            FILBUF
959 003400 000015          CR
960 003402 062703 000002          ADD      #2,            R3          ; SKIP OVER ADDRESS OF ASCII
961 003406 000313          SWAB    (R3),           (R4)+        ; FORMAT ASCII FOR MESSAGE
962 003410 112324          MOVVB   (R3)+,          (R4)+        ; PUT ASCII IN MESSAGE BUFFER
963 003412 112324          MOVVB   (R3)+,          (R4)+
964 003414 012705 006543 1$: MOV      #BRHEAD,       R5          ; GO PUT BOOT ROM HEADER IN MESS. BUF.
965 003420 004767 000754          JSR      PC,            FILBUF
966 003424 000015          CR
967 003426 112724 000015 2$: MOVVB   #CR,            (R4)+        ; PUT A CR/LF HERE
968 003432 112724 000012          MOVVB   #LF,            (R4)+
969 003436 005713          TST      (R3)          ; IS DATA EQUAL TO ZERO
970 003440 001464          BEG      5$

```



```

1026 003662 006000 1$: ROR RD ; IF ROM PRESENT C-BIT SETS
1027 003664 103406 BCS 2$ ; WHEN C-BITS SETS THATS "ONE"
1028 003666 005237 003252 INC 2#DAFLAC ; INCREMENT COUNTER-IF WE DO THIS
1029 003672 022737 000004 003252 CMP #4, 2#DAFLAG ; LOOP 4 TIMES, NO BOOT ROM PRESENT AT AL
1030 003700 003033 BGT SEQEX ; SO EXIT TEST
1031 003702 005700 2$: TST RD ; WE FOUND ONE, BUT IF RD IS EQUAL TO
1032 003704 001431 BEQ SEQEX ; ZERO THERE ONLY THE ONE SO EXIT
1033 003706 005037 003252 CLR 2#DAFLAG
1034 003712 012700 007420 MOV #BUF1, RD ; AT LEAST TWO START TEST, GET ROM DATA
1035 003716 032737 000001 001116 BIT #1, 2#ROMFIN ; IS DIAG ROM PRESENT
1036 003724 001402 BEQ 3$ ; IF NO DON'T UPDATE DATA TABLE ADDRESS
1037 003726 062700 000004 ADD #4, RD ; IF YES THAN UPDATE TABLE ADDRESS
1038 003732 004767 000040 3$: JSR PC, GETCOD ; GO GET A DEVICE CODE
1039 003736 005704 TST R4 ; UPON RETURN FROM GET COD R4 INDICATES
1040 003740 001013 BNE SEQEX ; IF VALID DEVICE CODE WAS FOUND
1041 003742 004767 000030 JSR PC, GETCOD ; CODES SC EXIT, IF NOT THEN GET NEXT COD
1042 003746 026767 000020 000020 CMP FRSCOD, LASCOD ; IS 1ST CODE ALPHABETICALLY LESS THE 2ND
1043 003754 003003 BGT 4$ ; IF NO SEQUENCE ERROR
1044 003756 005704 TST R4 ; IF R4 NOT EQUAL TO ZERO GET CON
1045 003760 001003 BNE SEQEX ; COULD NOT FIND VALID DEVICE CODE, SO EXIT
1046 003762 000763 BR 3$ ; IF NOT THEN GO GET NEXT DEVICE CODE
1047 003764 104401 4$: TYPE ; TYPE OUT WARNING MESSAGE
1048 003766 007317 SEQMSG
1049 003770 000207 SEQEX: RTS PC ; RETURN
1050
1051 003772 000000 FRSCOD: .WORD 0
1052 003774 000000 LASCOD: .WORD 0
1053
1054
1055
1056
1057
1058
1059
1060
1061

```

```

: GET DEVICE CODES SUBROUTINE
: THIS SUBROUTINE IS CALLED BY THE SEQUENCE TEST ROUTINE TO GET THE
: DEVICE CODES FROM THE SIZING BUFFER. THE ROUTINE VALIDATES EACH DEVICE
: CODE, THAT IS IT MAKES SURE THE CODE IS THE FIRST CODE IN A ROM. IF
: IT CAN NOT FIND A VALID DEVICE CODE IT MAKES R4 NONZERO BEFORE RETURNING.

```

```

1062 003776 005004 GETCOD: CLR R4
1063 004000 005737 003252 TST 2#DAFLAG ; IF DAFLAG IS ZERO PUT DEVICE CODE
1064 004004 001405 BEQ 1$ ; IN ERSCOD
1065 004006 012703 003774 MOV #LASCOD, R3 ; IF DAFLAG IS ONE PUT DEVICE CODE
1066 004012 005037 003252 CLR 2#DAFLAG ; IN LASCOD AND CLEAR DAFLAG
1067 004016 000404 BR 2$ ; GET OVER DAFLAG=ZERO SETUP
1068 004020 012703 003772 1$: MOV #FRSCOD, R3 ; PUT DEVICE CODE IN FRS COD
1069 004024 005237 003252 INC 2#DAFLAG ; MAKE DAFLAG NOT ZERO
1070 004030 005710 2$: TST (R0) ; DOES DEVICE CODE EXIST
1071 004032 001002 BNE 3$ ; IF YES CHECK IF VALID
1072 004034 005104 COM R4 ; IF NO MAKE R4 NON-ZERO
1073 004036 000410 BR GETEND ; AND EXIT
1074 004040 012001 3$: MOV (R0)+, R1 ; GET ADDRESS OF DEVICE CODE
1075 004042 032701 000170 BIT #170, R1 ; IF BIT 3-6 CLEAR THIS IS 1ST CODE
1076 004046 001403 BEQ 4$ ; OF A ROM SO GET IT
1077 004050 062700 000002 ADD #2, RD ; IF BITS SET UPDATE TO NEXT CODE
1078 004054 000765 BR 2$ ; AND VALIDATE IT
1079 004056 012013 4$: MOV (R0)+, (R3) ; GET DEVICE CODE AND STORE IT
1080 004060 000207 GETEND: RTS PC ; RETURN
1081

```

```

1082
1083
1084
1085
1086
1087
1088
1089          004062  104411
1090          004064  012704  010214
1091          004070  005237  001120
1092          004074  032777  002000  175002
1093          004102  001402
1094          004104  104401
1095          004106  006524
1096          004110  032777  020000  174766  1$:
1097          004116  001110
1098          004120  013700  001110
1099          004124  013767  001110  000174
1100          004132  012701  004362
1101          004136  000241
1102          004140  006000  2$:
1103          004142  103403
1104          004144  062701  000002
1105          004150  000773
1106          004152  011105  3$:
1107          004154  004767  000220
1108          004160  000015
1109          004162  032737  000023  001110
1110          004170  001424
1111          004172  032737  000020  001110
1112          004200  001415
1113          004202  013700  002460
1114          004206  012701  006512
1115          004212  000241
1116          004214  006000  4$:
1117          004216  103403
1118          004220  062701  000002
1119          004224  000773
1120          004226  112124  5$:
1121          004230  112124
1122          004232  000421
1123          004234  112324  6$:
1124          004236  112324
1125          004240  000416
1126          004242  016603  000006  7$:
1127          004246  004767  000170
1128          004252  004767  000122
1129          004256  000011
1130          004260  016603  000010
1131          004264  004767  000152
1132          004270  004767  000104
1133          004274  000011
1134          004276  112724  000015  8$:
1135          004302  112724  000012
1136          004306  112724  000000
1137          004312  005767  174640

```

```

:ERROR HANDLER ROUTINE
:THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
:OUT.
ERRHAN: SAVREG
MOV      #ERRMSG, R4
INC      @#ERRCNT
BIT      #BIT10, 1$
BEQ      1$
        ; INCREMENT ERROR COUNTER
        ; BELL ON ERROR
        ; BRANCH IF NO
        ; INHIBIT ERROR TYPEOUT
        ; GET ERROR CODE
        ; SAVE ERROR CODE FOR APT
        ; GET ERROR HEADER TABLE
        ; C-BIT USED TO STOP STEPPING THROUGH TAB
        ; ROTATE ERROR CODES
        ; IF C-BIT SET STOP STEPPING
        ; IF C-BIT CLEAR STEP TO NEXT HEADER
        ; GO STEP
        ; PUT HEADER ADDRESS IN REGISTER.
        ; GO PUT HEADER IN ERROR MESSAGE BUF.
        ; IS ERROR A CRC, APTER1 OR APTER2
        ; IF NO GO FORMAT APTER3, APTER4
        ; IS ERROR CRC
        ; IF NO FORMAT APTER1, APTER2
        ; GET ROM COUNTER
        ; GET ROM NUMBER TABLE
        ; C-BIT USED TO STOP STEPPING THROUGH TAB
        ; ROTATE ROM NUMBER
        ; IF C-BIT SET STOP STEPPING
        ; IF C-BIT CLEAR STEP TO NEXT HEADER
        ; CONTINUE STEPPING
        ; PUT ROM NUMBER IN ERROR MESSAGE BUF
        ; GO TO END OF ROUTINE
        ; PUT BAD DEVICE CODE IN ERROR MESS. BUF.
        ; GO TO END OF ROUTINE
        ; GET SAVED EXPECTED VALUE
        ; GO COVERT OCTAL TO ASCII
        ; GO PUT DATA IN ERROR MESSAGE BUF.
        ; GET SAVED RECEIVED VALUE
        ; GO COVERT OCTAL TO ASCII
        ; GO PUT DATA IN ERROR MESSAGE BUFFER.
        ; PUT CR/LF AT END OF MESSAGE
        ; PUT TERMINATOR AT END OF MESSAGE
        ; TST IF ON APT

```

```

1138 004316 001405          BEQ      11$          ; IF NO BRANCH
1139 004320 004767 000534  JSR      PC,          ; GO REPORT ERROR
1140 004324 010214          ERRMSG
1141 004326 000000          9$:      .WORD      0
1142 004330 000000          HALT
1143 004332 004767 000530  11$:     JSR      PC,          ; THEN HALT
1144 004336 010214          ERRMSG          ; IF NOT ON APT JUST REPORT ERROR
1145 004340 032777 100000 174536 10$:     BIT      #BIT15,    ; IS HALT ON ERROR SET
1146 004346 001401          BEQ      12$          ; IF NO SKIP OVER HALT
1147 004350 000000          HALT
1148 004352 005037 001110 12$:     CLR      @#ROMERR    ; CLEAR ERROR FLAGS
1149 004356 104412          RESREG
1150 004360 000207          RTS      PC
1151
1152 004362 006756          EHEADT: ER1MSG
1153 004364 006722          ER2MSG
1154 004366 007014          ER3MSG
1155 004370 007072          ER4MSG
1156 004372 007146          CRCMSG
1157 004374 007172          PFMSG
1158 004376 007250          NOROMM
1159
1160
1161
1162          ;FILL BUFFER ROUTINE
1163          ;THIS SUBROUTINE FILLS THE MESSAGE BUFFER WILL ASCII
1164          ;CHARACTERS.
1165
1166 004400 022776 000011 000000 FILBUF: CMP      #HT,          @ (R6)          ; IS FIRST CHARACTER A TAB OR CR
1167 004406 001003          BNE
1168 004410 112724 000011          MOVB     #HT,          (R4)+          ; IF CR THEN GO PUT CR/LF IN BUFFER
1169 004414 000404          BR
1170 004416 112724 000015 1$:      MOVB     #CR,          (R4)+          ; IF TAB THEN PUT TAB IN BUFFER
1171 004422 112724 000012          MOVB     #LF,          (R4)+          ; GET OVER NEXT LINE
1172 004426 112524 2$:      MOVB     (R5)+,        (R4)+          ; MOV CR/LF TO BUFFER
1173 004430 105715          TSTB     (R5)
1174 004432 001375          BNE
1175 004434 062716 000002          ADD      #2,          (R6)          ; PUT A CHARACTER IN MESSAGE BUFFER
1176 004440 000207          RTS      PC          ; IS NEXT CHARACTER ZERO
1177
1178
1179
1180
1181          ;OCTAL TO ASCII CONVERSION ROUTINE
1182          ;THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND
1183          ;CONVERTS IT TO 6 ASCII CHARACTERS
1184
1185 004442 012700 007504          OCASC:  MOV      #OCTBUF,    R0          ; GET BUFFER ADDRESS
1186 004446 005020          CLR      (R0)+          ; CLEAR BUFFER
1187 004450 005020          CLR      (R0)+
1188 004452 005020          CLR      (R0)+
1189 004454 005020          CLR      (R0)+
1190 004456 012700 007504          MOV      #OCTBUF,    R0          ; GET BUFFER ADDRESS
1191 004462 010337 004544          MOV      R3,          @#TEMP        ; GET OCTAL NUMBER
1192 004466 000241          CLC
1193 004470 006137 004544          ROL      @#TEMP        ; CLEAR CARRY
1194
1195
1196
1197
1198
1199
1200

```

1194 004474 006110
 1195 004476 152710 000060
 1196 004502 005200
 1197 004504 020027 007512
 1198 004510 001003
 1199 004512 012705 007504
 1200 004516 000207
 1201 004520 006137 004544
 1202 004524 106110
 1203 004526 006137 004544
 1204 004532 106110
 1205 004534 006137 004544
 1206 004540 106110
 1207 004542 000755
 1208 004544 000000

1\$: ROL (R0)
 BISB #60, (R0)
 INC RO
 CMP RO, #OCTBUF+6
 BNE 2\$
 MOV #OCTBUF, R5
 RTS PC
 2\$: ROL @TEMP
 ROLB (R0)
 ROL @TEMP
 ROLB (R0)
 ROL @TEMP
 ROLB (R0)
 BR 1\$
 TEMP: 0

: ROTATE CARRY BIT INTO BUFFER
 : MAKE IT ASCII
 : UPDATE BUFFER ADDRESS
 : HAVE WE CONVERTED ALL THE NUMBER
 : IF NO CONTINUE
 : IF YES PUT BUFFER ADDRESS IN REGISTER
 : RETURN
 : ROTATE BIT INTO CARRY BIT
 : ROTATE CARRY BIT INTO BUFFER
 :
 :
 : GO TO START OF LOOP

: THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
 : VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.

1215 004546 010346
 1216 004550 011303
 1217 004552 004767 177664
 1218 004556 012603
 1219 004560 000207

OCADD: MOV R3, -(R6) : SAVE THE VALUE OF R3
 MOV (R3), R3 : PUT THE DATA TO BE CONVERTED IN R3
 JSR PC, OCASC : GO CONVERT OCTAL TO ASCII
 MOV (R6)+, R3 : RESTORE R3
 RTS PC : RETURN

.SBTTL TYPE ROUTINE

: *****
 : *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 : *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 : *NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 : *NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 : *NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 : *
 : *CALL:
 : *1) USING A TRAP INSTRUCTION
 : * TYPE ,MESADR ;; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 : *OR
 : * TYPE
 : * MESADR
 : *

1242 004562 105767 000265
 1243 004566 100002
 1244 004570 000000
 1245 004572 000430
 1246 004574 010046
 1247 004576 017600 000002
 1248 004602 122767 000001 174346
 1249 004610 001011

\$TYPE: TSTB \$TPFLG : IS THERE A TERMINAL?
 BPL 1\$: BR IF YES
 HALT : HALT HERE IF NO TERMINAL
 BR 3\$: LEAVE
 1\$: MOV RO, -(SP) : SAVE RO
 MOV @2(SP), RO : GET ADDRESS OF ASCIZ STRING
 CMPB #APTENV, \$ENV : RUNNING IN APT MODE
 BNE 62\$: NO, GO CHECK FOR APT CONSOLE

1250	004612	132767	000100	174337	BITB	#APTSPool,\$ENVm	::SPOOL MESSAGE TO APT	
1251	004620	001405			BEQ	62\$::NO GO CHECK FOR CONSOLE	
1252	004622	010067	000004		MOV	RO,61\$::SETUP MESSAGE ADDRESS FOR APT	
1253	004626	004767	000234		JSR	PC,\$ATY3	::SPOOL MESSAGE TO APT	
1254	004632	000000			.WORD	0	::MESSAGE ADDRESS	
1255	004634	132767	000040	174315	BITB	#APTCSUP,\$ENVm	::APT CONSOLE SUPPRESSED	
1256	004642	001003			BNE	60\$::YES,SKIP TYPE OUT	
1257	004644	112046			MOVb	(RO)+,-(SP)	::PUSH CHARACTER TO BE TYPED ONTO STACK	
1258	004646	001005			BNE	4\$::BR IF IT ISN'T THE TERMINATOR	
1259	004650	005726			TST	(SP)+	::IF TERMINATOR POP IT OFF THE STACK	
1260	004652	012600			MOV	(SP)+,RO	::RESTORE RO	
1261	004654	062716	000002		ADD	#2,(SP)	::ADJUST RETURN PC	
1262	004660	000002			RTI		::RETURN	
1263	004662	122716	000011		CMPb	#HT,(SP)	::BRANCH IF <HT>	
1264	004666	001430			BEQ	8\$		
1265	004670	122716	000200		CMPb	#CRLF,(SP)	::BRANCH IF NOT <CRLF>	
1266	004674	001006			BNE	5\$		
1267	004676	005726			TST	(SP)+	::POP <CR><LF> EQUIV	
1268	004700	104401			TYPE		::TYPE A CR AND LF	
1269	004702	005055			\$CRLF			
1270	004704	105067	000130		CLRB	\$CHARCNT	::CLEAR CHARACTER COUNT	
1271	004710	000755			BR	2\$::GET NEXT CHARACTER	
1272	004712	004767	000056		JSR	PC,\$TYPEC	::GO TYPE THIS CHARACTER	
1273	004716	126726	000130		CMPb	\$FILLC,(SP)+	::IS IT TIME FOR FILLER CHARS.?	
1274	004722	001350			BNE	2\$::IF NO GO GET NEXT CHAR.	
1275	004724	016746	000120		MOV	\$NULL,-(SP)	::GET # OF FILLER CHARS. NEEDED	
1276							::AND THE NULL CHAR.	
1277	004730	105366	000001		DECB	1(SP)	::DOES A NULL NEED TO BE TYPED?	
1278	004734	002770			BLT	6\$::BR IF NO--GO POP THE NULL OFF OF STACK	
1279	004736	004767	000032		JSR	PC,\$TYPEC	::GO TYPE A NULL	
1280	004742	105367	000072		DECB	\$CHARCNT	::DO NOT COUNT AS A COUNT	
1281	004746	000770			BR	7\$::LOOP	
1282								
1283								
1284								
1285	004750	112716	000040		MOVb	#'(SP)	::REPLACE TAB WITH SPACE	
1286	004754	004767	000014		JSR	PC,\$TYPEC	::TYPE A SPACE	
1287	004760	132767	000007	000052	BITB	#7,\$CHARCNT	::BRANCH IF NOT AT	
1288	004766	001372			BNE	9\$::TAB STOP	
1289	004770	005726			TST	(SP)+	::POP SPACE OFF STACK	
1290	004772	000724			BR	2\$::GET NEXT CHARACTER	
1291	004774	105777	000044		\$TYPEC:	\$STPS	::WAIT UNTIL PRINTER IS READY	
1292	005000	100375			BPL	\$TYPEC		
1293	005002	116677	000002	000036	MOVb	2(SP),\$STPB	::LOAD CHAR TO BE TYPED INTO DATA REG.	
1294	005010	122766	000015	000002	CMPb	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?	
1295	005016	001003			BNE	1\$::BRANCH IF NO	
1296	005020	105067	000014		CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT	
1297	005024	000406			BR	\$TYPEX	::EXIT	
1298	005026	122766	000012	000002	1\$:	CMPb	#LF,2(SP)	::IS CHARACTER A LINE FEED?
1299	005034	001402			BEQ	\$TYPEX	::BRANCH IF YES	
1300	005036	105227			INCB	(PC)+	::COUNT THE CHARACTER	
1301	005040	000000			\$CHARCNT:	.WORD	::CHARACTER COUNT STORAGE	
1302	005042	000207			\$TYPEX:	RTS		
1303								
1304	005044	177564			\$STPS:	.WORD	177564	::TTY PRINTER STATUS REG. ADDRESS
1305	005046	177566			\$STPB:	.WORD	177566	::TTY PRINTER BUFFER REG. ADDRESS

```

1306 005050 000 $NULL: .BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
1307 005051 002 $FILLS: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1308 005052 012 $FILLC: .BYTE 12 ;;INSERT FILL CHARS. AFTER A "LINE FEED"
1309 005053 000 $TPFLG: .BYTE 0 ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1310 005054 077 $QUES: .ASCII "?" ;;QUESTION MARK
1311 005055 015 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
1312 005056 000012 $LF: .ASCIZ <12> ;;LINEFEED
1313 .SBTTL APT COMMUNICATIONS ROUTINE
1314
1315 *****
1316 005060 112767 000001 000236 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
1317 005066 112767 000001 000226 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
1318 005074 000403 BR $ATYC
1319 005076 112767 000001 000220 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
1320 005104 $ATYC:
1321 005104 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
1322 005106 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
1323 005110 105767 000206 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
1324 005114 001450 BEQ 5$ ;;IF NOT: BR
1325 005116 122767 000001 174032 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
1326 005124 001031 BNE 3$ ;;IF NOT: BR
1327 005126 132767 000100 174023 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1328 005134 001425 BEQ 3$ ;;IF NOT: BR
1329 005136 017600 MOV #4(SP),RO ;;GET MESSAGE ADDR.
1330 005142 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
1331 005150 005767 173762 1$ TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
1332 005154 001375 BNE 1$ ;;IF NOT: WAIT
1333 005156 010067 173770 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
1334 005162 105720 2$ TSTB (RO)+ ;;FIND END OF MESSAGE
1335 005164 001376 BNE 2$
1336 005166 166700 173760 SUB $MSGAD,RO ;;SUB START OF MESSAGE
1337 005172 006200 ASR RO ;;GET MESSAGE LNTH IN WORDS
1338 005174 010067 173754 MOV RO,$MSGGLT ;;PUT LENGTH IN MAILBOX
1339 005200 012767 000004 173730 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
1340 005206 000413 BR 5$
1341 005210 017667 000004 000016 3$ MOV #4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
1342 005216 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
1343 005224 016746 172546 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
1344 005230 004767 177326 JSR PC,$TYPE ;;CALL TYPE MACRO
1345 005234 000000 4$ .WORD 0
1346 005236 5$
1347 005236 105767 000062 10$ TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
1348 005242 001416 BEQ 12$ ;;IF NOT: BR
1349 005244 005767 173706 TST $ENV ;;RUNNING UNDER APT?
1350 005250 001413 BEQ 12$ ;;IF NOT: BR
1351 005252 005767 173660 11$ TST $MSGTYPE ;;FINISHED LAST MESSAGE?
1352 005256 001375 BNE 11$ ;;IF NOT: WAIT
1353 005260 017667 000004 173652 MOV #4(SP),$FATAL ;;GET ERROR #
1354 005266 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
1355 005274 005267 173636 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
1356 005300 105067 000020 12$ CLRB $FFLG ;;CLEAR FATAL FLAG
1357 005304 105067 000013 CLRB $LFLG ;;CLEAR LOG FLAG
1358 005310 105067 000006 CLRB $MFLG ;;CLEAR MESSAGE FLAG
1359 005314 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
1360 005316 012600 MOV (SP)+,RO ;;POP STACK INTO RO
1361 005320 000207 RTS PC ;;RETURN

```

```

1362 005322 000 SMFLG: .BYTE 0 ;; MESSG. FLAG
1363 005323 000 SLFLG: .BYTE 0 ;; LOG FLAG
1364 005324 000 SFFLG: .BYTE 0 ;; FATAL FLAG
1365 005326 .EVEN
1366 000200 APTSIZE=200
1367 000001 APTENV=001
1368 000100 APTSPool=100
1369 000040 APTCSUP=040
1370 .SBTTL TTY INPUT ROUTINE
1371
1372 *****
1373 005326 177560 $TKS: .WORD 177560 ;; TTY KBD STATUS
1374 005330 177562 $TKB: .WORD 177562 ;; TTY KBD BUFFER
1375 .ENABL LSB
1376
1377 *****
1378 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1379 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1380 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1381 *WHEN OPERATING IN TTY FLAG MODE.
1382 005332 022767 000176 173544 $CKSWR: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
1383 005340 001074 BNE 15$ ;; BRANCH IF NO
1384 005342 105777 177760 TSTB $TKS ;; CHAR THERE?
1385 005346 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
1386 005350 117746 177754 MOVB $TKB,-(SP) ;; SAVE THE CHAR
1387 005354 042716 177600 BIC #1C177,(SP) ;; STRIP-OFF THE ASCII
1388 005360 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
1389 005364 001062 BNE 15$ ;; NO, RETURN TO USER
1390 005366 126727 173506 000001 CMPB SAUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
1391 005374 001456 BEQ 15$ ;; BRANCH IF YES
1392
1393 005376 104401 006057 $GTSWR: TYPE ,SCNTLG ;; ECHO THE CONTROL-G (↑G)
1394 005402 104401 006064 TYPE ,SMSWR ;; TYPE CURRENT CONTENTS
1395 005406 016746 172564 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
1396 005412 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
1397 005414 104401 006075 TYPE ,SMNEW ;; PROMPT FOR NEW SWR
1398 005420 005046 19$: CLR -(SP) ;; CLEAR COUNTER
1399 005422 005046 CLR -(SP) ;; THE NEW SWR
1400 005424 105777 177676 7$: TSTB $TKS ;; CHAR THERE?
1401 005430 100375 BPL 7$ ;; IF NOT TRY AGAIN
1402
1403 005432 117746 177672 MOVB $TKB,-(SP) ;; PICK UP CHAR
1404 005436 042716 177600 BIC #1C177,(SP) ;; MAKE IT 7-BIT ASCII
1405
1406
1407
1408 005442 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
1409 005446 001005 BNE 10$ ;; BRANCH IF NOT
1410 005450 104401 006052 TYPE ,SCNTLU ;; YES, ECHO CONTROL-U (↑U)
1411 005454 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
1412 005460 000757 BR 19$ ;; LET'S TRY IT AGAIN
1413
1414
1415 005462 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
1416 005466 001022 BNE 16$ ;; BRANCH IF NO
1417 005470 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?

```

1418	005474	001403				BEQ	11\$:: BRANCH IF YES
1419	005476	016677	000002	173400		MOV	2(SP), 2SWR	:: SAVE NEW SWR
1420	005504	062706	000006		11\$:	ADD	#6, SP	:: CLEAR UP STACK
1421	005510	104401	005055		14\$:	TYPE	\$CRLF	:: ECHO <CR> AND <LF>
1422	005514	126727	173361	000001		CMPB	\$INTAG, #1	:: RE-ENABLE TTY KBD INTERRUPTS?
1423	005522	001003				BNE	15\$:: BRANCH IF NOT
1424	005524	012777	000100	177574		MOV	#100, 2STKS	:: RE-ENABLE TTY KBD INTERRUPTS
1425	005532	000002			15\$:	RTI		:: RETURN
1426	005534	004767	177234		16\$:	JSR	PC, \$TYPEC	:: ECHO CHAR
1427	005540	021627	000060			CMP	(SP), #60	:: CHAR < 0?
1428	005544	002420				BLT	18\$:: BRANCH IF YES
1429	005546	021627	000067			CMP	(SP), #67	:: CHAR > ??
1430	005552	003015				BGT	18\$:: BRANCH IF YES
1431	005554	042726	000060			BIC	#60, (SP)+	:: STRIP-OFF ASCII
1432	005560	005766	000002			TST	2(SP)	:: IS THIS THE FIRST CHAR
1433	005564	001403				BEQ	17\$:: BRANCH IF YES
1434	005566	006316				ASL	(SP)	:: NO, SHIFT PRESENT
1435	005570	006316				ASL	(SP)	:: CHAR OVER TO MAKE
1436	005572	006316				ASL	(SP)	:: ROOM FOR NEW ONE.
1437	005574	005266	000002		17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
1438	005600	056616	177776			BIS	-2(SP), (SP)	:: SET IN NEW CHAR
1439	005604	000707				BR	7\$:: GET THE NEXT ONE
1440	005606	104401	005054		18\$:	TYPE	\$QUES	:: TYPE ?<CR><LF>
1441	005612	000720				BR	20\$:: SIMULATE CONTROL-U
1442						.DSABL	LSB	

1442								:: *****
1443								:: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1444								:: *CALL:
1445								:: *
1446								:: *RDCHR
1447								:: * INPUT A SINGLE CHARACTER FROM THE TTY
1448								:: * RETURN HERE
1449								:: * CHARACTER IS ON THE STACK
1450								:: * WITH PARITY BIT STRIPPED OFF
1451								:: *
1452								:: *
1453	005614	011646			\$RDCHR:	MOV	(SP), -(SP)	:: PUSH DOWN THE PC
1454	005616	016666	000004	000002		MOV	4(SP), 2(SP)	:: SAVE THE PS
1455	005624	105777	177476		1\$:	TSTB	2STKS	:: WAIT FOR
1456	005630	100375				BPL	1\$:: A CHARACTER
1457	005632	117766	177472	000004		MOVB	2STKB, 4(SP)	:: READ THE TTY
1458	005640	042766	177600	000004		BIC	#1C<177>, 4(SP)	:: GET RID OF JUNK IF ANY
1459	005646	026627	000004	000023		CMP	4(SP), #23	:: IS IT A CONTROL-5?
1460	005654	001013				BNE	3\$:: BRANCH IF NO
1461	005656	105777	177444		2\$:	TSTB	2STKS	:: WAIT FOR A CHARACTER
1462	005662	100375				BPL	2\$:: LOOP UNTIL ITS THERE
1463	005664	117746	177440			MOVB	2STKB, -(SP)	:: GET CHARACTER
1464	005670	042716	177600			BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
1465	005674	022627	000021			CMP	(SP)+, #21	:: IS IT A CONTROL-Q?
1466	005700	001366				BNE	2\$:: IF NOT DISCARD IT
1467	005702	000750				BR	1\$:: YES, RESUME
1468	005704	026627	000004	000140	3\$:	CMP	4(SP), #140	:: IS IT UPPER CASE?
1469	005712	002407				BLT	4\$:: BRANCH IF YES
1470	005714	026627	000004	000175		CMP	4(SP), #175	:: IS IT A SPECIAL CHAR?
1471	005722	003003				BGT	4\$:: BRANCH IF YES
1472	005724	042766	000040	000004		BIC	#40, 4(SP)	:: MAKE IT UPPER CASE
1473	005732	000002			4\$:	RTI		:: GO BACK TO USER

```

1474      ;:*****
1475      ;:THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1476      ;:CALL:
1477      ;:      RDLIN          ;: INPUT A STRING FROM THE TTY
1478      ;:      RETURN HERE   ;: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1479      ;:                  ;: TERMINATOR WILL BE A BYTE OF ALL 0'S
1480
1481 005734 010346 $RDLIN: MOV      R3, -(SP)      ;: SAVE R3
1482 005736 012703 1$:      MOV      #STTYIN, R3    ;: GET ADDRESS
1483 005742 022703 2$:      CMP      #STTYIN+8., R3  ;: BUFFER FULL?
1484 005746 101405      BLOS     4$              ;: BR IF YES
1485 005750 104407      RDCHR    ;: GO READ ONE CHARACTER FROM THE TTY
1486 005752 112613      MOVB    (SP)+, (R3)      ;: GET CHARACTER
1487 005754 122713 10$:    CMPB    #177, (R3)      ;: IS IT A RUBOUT
1488 005760 001003      BNE     3$              ;: SKIP IF NOT
1489 005762 104401 4$:     TYPE    $QUES        ;: TYPE A '?'
1490 005766 000763      BR      1$              ;: CLEAR THE BUFFER AND LOOP
1491 005770 111367 3$:     MOVB    (R3), 9$      ;: ECHO THE CHARACTER
1492 005774 104401      TYPE    9$
1493 006000 122723      CMPB    #15, (R3)+      ;: CHECK FOR RETURN
1494 006004 001356      BNE     2$              ;: LOOP IF NOT RETURN
1495 006006 105063 177777 CLRB    -1(R3)      ;: CLEAR RETURN (THE 15)
1496 006012 104401      TYPE    $LF          ;: TYPE A LINE FEED
1497 006016 012603      MOV     (SP)+, R3      ;: RESTORE R3
1498 006020 011646      MOV     (SP)-, (SP)    ;: ADJUST THE STACK AND PUT ADDRESS OF THE
1499 006022 016666 000004 000002 MOV     4(SP), 2(SP)  ;: FIRST ASCII CHARACTER ON IT
1500 006030 012766 006042 000004 MOV     #STTYIN, 4(SP)
1501 006036 000002      RTI          ;: RETURN
1502 006040 000      9$:     .BYTE    0          ;: STORAGE FOR ASCII CHAR. TO TYPE
1503 006041 000      .BYTE    0          ;: TERMINATOR
1504 006042 000010 $TTYIN: .BLKB    8.    ;: RESERVE 8 BYTES FOR TTY INPUT
1505 006052 052536 005015 000 $CNTLU: .ASCIZ  /↑U/<15><12> ;: CONTROL "U"
1506 006057 136 006507 000012 $CNTLG: .ASCIZ  /↑G/<15><12> ;: CONTROL "G"
1507 006064 005015 053523 020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
1508 006072 020075 000
1509 006075 040 047040 053505 $MNEW:  .ASCIZ  / NEW = /
1510 006102 036440 000040
1511      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1512
1513      ;:*****
1514      ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1515      ;:OCTAL (ASCII) NUMBER AND TYPE IT.
1516      ;:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1517      ;:CALL:
1518      ;:      MOV      NUM, -(SP)      ;: NUMBER TO BE TYPED
1519      ;:      TYPOS    ;: CALL FOR TYPEOUT
1520      ;:      .BYTE    N          ;: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1521      ;:      .BYTE    M          ;: M=1 OR 0
1522      ;:                  ;: 1=TYPE LEADING ZEROS
1523      ;:                  ;: 0=SUPPRESS LEADING ZEROS
1524
1525      ;:$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1526      ;:$TYPOS OR $TYPOC
1527      ;:CALL:
1528      ;:      MOV      NUM, -(SP)      ;: NUMBER TO BE TYPED
1529      ;:      TYPON   ;: CALL FOR TYPEOUT

```

```

1530      ;*
1531      ;*$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1532      ;*CALL:
1533      ;*      MOV      NUM,-(SP)      ;:NUMBER TO BE TYPED
1534      ;*      TYPOC      ;:CALL FOR TYPEOUT
1535
1536 006106 017646 000000      $STYPOC: MOV      2(SP),-(SP)      ;:PICKUP THE MODE
1537 006112 116667 000001 000211      MOVVB     1(SP),%SOFILL ;:LOAD ZERO FILL SWITCH
1538 006120 112667 000207      MOVVB     (SP)+,%SOMODE+1 ;:NUMBER OF DIGITS TO TYPE
1539 006124 062716 000002      ADD       #2,(SP)      ;:ADJUST RETURN ADDRESS
1540 006130 000406      BR        $STYPOC
1541 006132 112767 000001 000171      $STYPOC: MOVVB     #1,%SOFILL ;:SET THE ZERO FILL SWITCH
1542 006140 112767 000006 000165      MOVVB     #6,%SOMODE+1 ;:SET FOR SIX(6) DIGITS
1543 006146 112767 000005 000154      $STYPOC: MOVVB     #5,%SOCNT ;:SET THE ITERATION COUNT
1544 006154 010346      MOV       R3,-(SP)      ;:SAVE R3
1545 006156 010446      MOV       R4,-(SP)      ;:SAVE R4
1546 006160 010546      MOV       R5,-(SP)      ;:SAVE R5
1547 006162 116704 000145      MOVVB     %SOMODE+1,R4 ;:GET THE NUMBER OF DIGITS TO TYPE
1548 006166 005404      NEG       R4
1549 006170 062704 000006      ADD       #6,R4      ;:SUBTRACT IT FOR MAX. ALLOWED
1550 006174 110467 000132      MOVVB     R4,%SOMODE ;:SAVE IT FOR USE
1551 006200 116704 000125      MOVVB     %SOFILL,R4 ;:GET THE ZERO FILL SWITCH
1552 006204 016605 000012      MOV       12(SP),R5 ;:PICKUP THE INPUT NUMBER
1553 006210 005003      CLR       R3      ;:CLEAR THE OUTPUT WORD
1554 006212 006105      1$:      ROL       R5      ;:ROTATE MSB INTO "C"
1555 006214 000404      BR        3$
1556 006216 006105      2$:      ROL       R5      ;:GO DO MSB
1557 006220 006105      ROL       R5      ;:FORM THIS DIGIT
1558 006222 006105      ROL       R5
1559 006224 010503      MOV       R5,R3
1560 006226 006103      3$:      ROL       R3      ;:GET LSB OF THIS DIGIT
1561 006230 105367 000076      DECB     %SOMODE ;:TYPE THIS DIGIT?
1562 006234 100016      BPL      7$      ;:BR IF NO
1563 006236 042703 177770      BIC     #177770,R3 ;:GET RID OF JUNK
1564 006242 001002      BNE     4$      ;:TEST FOR 0
1565 006244 005704      TST     R4      ;:SUPPRESS THIS 0?
1566 006246 001403      BEQ     5$      ;:BR IF YES
1567 006250 005204      4$:      INC       R4      ;:DON'T SUPPRESS ANYMORE 0'S
1568 006252 052703 000060      BIS     #'0,R3 ;:MAKE THIS DIGIT ASCII
1569 006256 052703 000040      5$:      BIS     #' ,R3 ;:MAKE ASCII IF NOT ALREADY
1570 006262 110367 000040      MOVVB     R3,%S      ;:SAVE FOR TYPING
1571 006266 104401 006326      TYPE     #S      ;:GO TYPE THIS DIGIT
1572 006272 105367 000032      7$:      DECB     %SOCNT ;:COUNT BY 1
1573 006276 003347      BGT     2$      ;:BR IF MORE TO DO
1574 006300 002402      BLT     6$      ;:BR IF DONE
1575 006302 005204      INC     R4      ;:INSURE LAST DIGIT ISN'T A BLANK
1576 006304 000744      BR        2$      ;:GO DO THE LAST DIGIT
1577 006306 012605      6$:      MOV      (SP)+,R5 ;:RESTORE R5
1578 006310 012604      MOV      (SP)+,R4 ;:RESTORE R4
1579 006312 012603      MOV      (SP)+,R3 ;:RESTORE R3
1580 006314 016666 000002 000004      MOV      2(SP),4(SP) ;:SET THE STACK FOR RETURNING
1581 006322 012616      MOV      (SP)+,(SP)
1582 006324 000002      RTI
1583 006326 000      8$:      .BYTE   0 ;:RETURN
1584 006327 000      .BYTE   0 ;:STORAGE FOR ASCII DIGIT
1585 006330 000      $SOCNT: .BYTE   0 ;:TERMINATOR FOR TYPE ROUTINE
           ;:OCTAL DIGIT COUNTER

```

E03

CZM9BA0 M9312 BOOT TERMR BK
CZM9BA.P11 13-MAR-78 09:58

MACY11 30A(1052) 13-MAR-78 09:59 PAGE 30
BINARY TO OCTAL (ASCII) AND TYPE

SEQ 0030

1586 006331 000
1587 006332 000000
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605 006334
1606 006334 010046
1607 006336 010146
1608 006340 010246
1609 006342 010346
1610 006344 010446
1611 006346 010546
1612 006350 016646 000022
1613 006354 016646 000022
1614 006360 016646 000022
1615 006364 016646 000022
1616 006370 000002
1617
1618
1619
1620
1621 006372
1622 006372 012666 000022
1623 006376 012666 000022
1624 006402 012666 000022
1625 006406 012666 000022
1626 006412 012605
1627 006414 012604
1628 006416 012603
1629 006420 012602
1630 006422 012601
1631 006424 012600
1632 006426 000002
1633
1634
1635
1636
1637
1638
1639
1640
1641 006430 010046

\$OFILL: .BYTE 0 ;:ZERO FILL SWITCH
\$OMODE: .WORD 0 ;:NUMBER OF DIGITS TO TYPE
.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

;;*****
;*SAVE RO-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0

\$SAVREG:
MOV RO,-(SP) ;:PUSH RO ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
MOV R5,-(SP) ;:PUSH R5 ON STACK
MOV 22(SP),-(SP) ;:SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;:SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;:SAVE PS OF CALL
MOV 22(SP),-(SP) ;:SAVE PC OF CALL
RTI

;*RESTORE RO-R5
;*CALL:
;* RESREG

\$RESREG:
MOV (SP)+,22(SP) ;:RESTORE PC OF CALL
MOV (SP)+,22(SP) ;:RESTORE PS OF CALL
MOV (SP)+,22(SP) ;:RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;:RESTORE PS OF MAIN FLOW
MOV (SP)+,R5 ;:POP STACK INTO R5
MOV (SP)+,R4 ;:POP STACK INTO R4
MOV (SP)+,R3 ;:POP STACK INTO R3
MOV (SP)+,R2 ;:POP STACK INTO R2
MOV (SP)+,R1 ;:POP STACK INTO R1
MOV (SP)+,R0 ;:POP STACK INTO R0

.SBTTL TRAP DECODER

;;*****
;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.

\$TRAP: MOV RO,-(SP) ;:SAVE RO

```

1642 006432 016600 000002          MOV      2(SP),RO          ;;GET TRAP ADDRESS
1643 006436 005740                TST      -(RO)           ;;BACKUP BY 2
1644 006440 111000                MOV      (RO),RO         ;;GET RIGHT BYTE OF TRAP
1645 006442 006300                ASL      RO              ;;POSITION FOR INDEXING
1646 006444 016000 006464          MOV      $TRPAD(RO),RO   ;;INDEX TO TABLE
1647 006450 000200                RTS      RO              ;;GO TO ROUTINE
1648
1649
1650                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1651
1652 006452 011646 000004 000002 $TRAP2: MOV      (SP),-(SP) ;;MOVE THE PC DOWN
1653 006454 016666                MOV      4(SP),2(SP)    ;;MOVE THE PSW DOWN
1654 006462 000002                RTI                    ;;RESTORE THE PSW
1655
1656                                .SBTTL  TRAP TABLE
1657
1658                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1659                                ;*BY THE "TRAP" INSTRUCTION.
1660
1661                                ;          ROUTINE
1662                                ;-----
1663 006464 006452          $TRPAD: .WORD  $TRAP2
1664 006466 004562          $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
1665 006470 006132          $TYPOC ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1666 006472 006106          $TYPOS ;;CALL=TYPOS    TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
1667 006474 006146          $TYPON ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
1668
1669 006476 005402          $GTSWR ;;CALL=GTSWR    TRAP+5(104405) GET SOFT-SWR SETTING
1670
1671 006500 005332          $CKSWR ;;CALL=CKSWR    TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
1672 006502 005614          $RDCHR ;;CALL=RDCHR    TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
1673 006504 005734          $RDLIN ;;CALL=RDLIN     TRAP+10(104410) TTY TYPEIN STRING ROUTINE
1674 006506 006334          $$AVREG ;;CALL=SAVREG    TRAP+11(104411) SAVE RO-R5 ROUTINE
1675 006510 006372          $RESREG ;;CALL=RESREG  TRAP+12(104412) RESTORE RO-R5 ROUTINE
1676
1677
1678                                ;MESSAGES
1679
1680
1681 006512 030062          ROMNUM: .ASCII  /20/
1682 006514 032463          .ASCII  /35/
1683 006516 031463          .ASCII  /33/
1684 006520 032063          .ASCII  /34/
1685 006522 031063          .ASCII  /32/
1686 006524 177607 000377          BELL:   .ASCIZ  <207><377><377>
1687
1688
1689 006530 044504 043501 020056          DRHEAD: .ASCIZ  /DIAG. ROM /
1690 006536 047522 020115 000
1691 006543 102 047517 051524          BRHEAD: .ASCII  /BOOTSTRAP ROM ENTRY POINTS AND DEVICE CODES/<15><12>
1692 006550 051124 050101 051040
1693 006556 046517 042440 052116
1694 006564 054522 050040 044517
1695 006572 052116 020123 047101
1696 006600 020104 042504 044526
1697 006606 042503 041440 042117

```

1698	006614	051505	005015					
1699	006620	047011	020117	044504	.ASCIZ /	NO DIAG.	RUN DIAG.	DEVICE CODE/
1700	006626	043501	004456	052522				
1701	006634	020116	044504	043501				
1702	006642	004456	042504	044526				
1703	006650	042503	041440	042117				
1704	006656	000105						
1705	006660	051520	052505	047504	PFHEAD: .ASCIZ /	PSEUDO POWER-FAIL VECTOR ADDRESS /		
1706	006666	050040	053517	051105				
1707	006674	043055	044501	020114				
1708	006702	042526	052103	051117				
1709	006710	040440	042104	042522				
1710	006716	051523	000040					
1711	006722	047503	046125	020104	ER2MSG: .ASCIZ /	COULD NOT FIND DEVICE CODE /		
1712	006730	047516	020124	044506				
1713	006736	042116	042040	053105				
1714	006744	041511	020105	047503				
1715	006752	042504	000040					
1716	006756	047506	047125	020104	ER1MSG: .ASCIZ /	FOUND UNEXPECTED DEVICE CODE /		
1717	006764	047125	054105	042520				
1718	006772	052103	042105	042040				
1719	007000	053105	041511	020105				
1720	007006	047503	042504	000040				
1721	007014	047520	042527	026522	ER3MSG: .ASCII /	POWER-FAIL VECTOR ERROR/<15><12>		
1722	007022	040506	046111	053040				
1723	007030	041505	047524	020122				
1724	007036	051105	047522	006522				
1725	007044	012						
1726	007045	011	054105	042520	.ASCIZ /	EXPECTED	RECIEVED/<15><12>	
1727	007052	052103	042103	051011				
1728	007060	041505	042511	042526				
1729	007066	006504	000012					
1730	007072	047520	042527	026522	ER4MSG: .ASCII /	POWER-FAIL DATA ERROR/<15><12>		
1731	007100	040506	046111	042040				
1732	007106	052101	020101	051105				
1733	007114	047522	006522	012				
1734	007121	011	054105	042520	.ASCIZ /	EXPECTED	RECIEVED/<15><12>	
1735	007126	052103	042105	051011				
1736	007134	041505	042511	042526				
1737	007142	006504	000012					
1738	007146	051103	020103	051105	CRCMSG: .ASCIZ /	CRC ERROR IN ROM E-/		
1739	007154	047522	020122	047111				
1740	007162	051040	046517	042440				
1741	007170	000055						
1742	007172	047503	046125	020104	PFMSG: .ASCIZ /	COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/		
1743	007200	047516	020124	042504				
1744	007206	042524	046522	047111				
1745	007214	020105	047520	042527				
1746	007222	026522	040506	046111				
1747	007230	053040	041505	047524				
1748	007236	020122	042101	051104				
1749	007244	051505	000123					
1750	007250	047516	051040	046517	NOROMM: .ASCII /	NO ROMS TEST ERROR/<15><12>		
1751	007256	020123	042524	052123				
1752	007264	042440	051122	051117				
1753	007272	005015						

ABASE = 000000	506	547	
ACDW1 = 000000	506	549	
ACDW2 = 000000	506	550	
ACPUOP = 000000	506	521	
ADDW0 = 000000	506	551	
ADDW1 = 000000	506	552	
ADDW10 = 000000	506	561	
ADDW11 = 000000	506	562	
ADDW12 = 000000	506	563	
ADDW13 = 000000	506	564	
ADDW14 = 000000	506	565	
ADDW15 = 000000	506	566	
ADDW2 = 000000	506	553	
ADDW3 = 000000	506	554	
ADDW4 = 000000	506	555	
ADDW5 = 000000	506	556	
ADDW6 = 000000	506	557	
ADDW7 = 000000	506	558	
ADDW8 = 000000	506	559	
ADDW9 = 000000	506	560	
ADEVCT = 000000	506	512	
ADEVM = 000000	506	548	
AENV = 000000	506	517	
AENVM = 000000	506	518	
AFATAL = 000000	506	509	
AMADR1 = 000000	506	534	
AMADR2 = 000000	506	538	
AMADR3 = 000000	506	541	
AMADR4 = 000000	506	544	
AMAMS1 = 000000	506	528	
AMAMS2 = 000000	506	536	
AMAMS3 = 000000	506	539	
AMAMS4 = 000000	506	542	
AMSGAD = 000000	506	514	
AMSGLG = 000000	506	515	
AMSGTY = 00LJ00	506	508	
AMTYP1 = 000000	506	529	
AMTYP2 = 000000	506	537	
AMTYP3 = 000000	506	540	
AMTYP4 = 000000	506	543	
APASS = 000000	506	511	
APRIOR = 000000	506		
APTCSU = 000040	1255	1369#	
APTENV = 000001	1248	1325	1367#
APTER1 = 000001	434#	826	
APTER2 = 000002	435#	840	
APTER3 = 000004	436#	851	
APTER4 = 000010	437#	857	
APTSIZ = 000200	597	1366#	
APTSP0 = 000100	1250	1327	1368#
ASWREG = 000000	506	519	
ATESTN = 000000	506	510	
AUNIT = 000000	506	513	
AUSWR = 000J00	506	520	
AVECT1 = 000000	506	545	
AVECT2 = 000000	506	546	

SCDH1	001216	549#	853					
SCDH2	001220	550#						
SCHARC	005040	1270*	1280*	1287	1296*	1301#		
SCKSWR	005332	1382#	1671					
SCMTAG=	***** U	574	578					
SCNTLG	006057	1393	1506#					
SCNTLU	006052	1410	1505#					
SCPUOP	001164	521#						
SCRLF	005055	1269	1311#	1421	1505			
SDDW0	001222	551#	818	831				
SDDW1	001224	552#						
SDDW10	001246	561#						
SDDW11	001250	562#						
SDDW12	001252	563#						
SDDW13	001254	564#						
SDDW14	001256	565#						
SDDW15	001260	566#						
SDDW2	001226	553#						
SDDW3	001230	554#						
SDDW4	001232	555#						
SDDW5	001234	556#						
SDDW6	001236	557#						
SDDW7	001240	558#						
SDDW8	001242	559#						
SDDW9	001244	560#						
SDEVCT	001146	512#						
SDEVN	001214	548#						
SDOAGN	002162	678	685	691#				
SENDAD	002152	464	605	687#				
SENDCT	002126	579	680#					
SENDMG	002171	682	695#					
SENULL	002166	683	694#					
SENV	001156	517#	611	1137	1248	1325	1349	
SEVM	001157	518#	597	802	1250	1255	1327	
SEOP	002102	656	659	672#				
SEOPCT	002120	579*	677#	681				
SETABL	001156	516#						
SETEND	001262	502	569#					
SFATAL	001140	509#	1353*					
SFFLG	005324	1316*	1319*	1347	1356*	1364#		
SFILLC	005052	1273	1308#					
SFILLS	005051	1307#						
SGET42	002142	684#						
SGTSWR	005402	1394#	1669					
SHD =	000003	312	313					
SHIBTS	001122	497#						
SINTAG	001101	471#	1422	1511				
SLF	005056	1312#	1496	1505				
SLFLG	005323	1357*	1363#					
SMADR1	001170	534#						
SMADR2	001174	538#						
SMADR3	001200	541#						
SMADR4	001204	544#						
SMAIL	001136	498	502	507#	596	611	1248	
SMAMS1	001166	528#						
SMAMS2	001172	536#						

004

CZM9BA0 M9312 BOOT TERM BK
CZM9BA.P11 13-MAR-78 09:58

MACY11 30A(1052) 13-MAR-78 09:59 PAGE 44
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0042

. ABS. 010314 000

ERRORS DETECTED: 0

CZM9BA,CZM9BA/SOL/CRF/NL:TOC=CZM9BA.P11
RUN-TIME: 20 5 .7 SECONDS
RUN-TIME RATIO: 97/27=3.5
CORE USED: 20K (39 PAGES)

E04