

# KD11-K

11/6X SERIES CPU EXER  
MD-11-DQKDC-A

EP-DQKDC-A-DL-A  
COPYRIGHT © 1977

APR 1977  
**digital**  
FICHE 1 OF 2  
MADE IN USA

This microfiche card contains a grid of 144 frames of technical data, organized into 12 rows and 12 columns. The data is presented in a structured, tabular format, likely representing a memory dump or a series of diagnostic test results. Each frame contains multiple columns of alphanumeric characters, including hexadecimal and decimal values, and some frames include headers or labels. The overall layout is consistent across the grid, with each frame providing a snapshot of system information at a specific point in time or for a specific component.



# KD11-K

11/6X SERIES CPU EXER  
MD-11-DQKDC-A

EP-DQKDC-A-DL-A  
COPYRIGHT © 1977  
FICHE 2 OF 2

APR 1977  
**digital**  
MADE IN USA

The microfiche card contains 48 frames of data, arranged in 12 rows and 4 columns. Each frame displays a different portion of a program or data set, consisting of lines of alphanumeric characters. The text is too small to be legible in this image, but it appears to be a standard format for a program listing or data dump from a mainframe computer system.

11/6X



B01

.REM 2

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

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DQKDC-A-D  
PRODUCT NAME: 11/6X SERIES CPU EXERCISER  
DATE CREATED: JANUARY 24, 1977  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: 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 MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1977 BY DIGITAL EQUIPMENT CORPORATION



CONTENTS

45  
46  
47  
48  
49  
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

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 STORAGE
  - 2.3 PRELIMINARY PROGRAMS
- 3.0 LOADING PROCEDURE
  - 3.1 METHOD
- 4.0 STARTING PROCEDURE
  - 4.1 CONTROL SWITCH SETTINGS
  - 4.2 STARTING ADDRESSES
  - 4.3 PROGRAM AND OPERATOR ACTION
- 5.0 OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 DISPLAY REGISTER
  - 5.3 OPERATOR ACTION
- 6.0 ERRORS
  - 6.1 ERROR HALTS AND DESCRIPTION
  - 6.2 ERROR RECOVERY
- 7.0 WARNINGS AND EXCEPTIONS
  - 7.1 WARNINGS
  - 7.2 EXCEPTIONS
- 8.0 MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 PASS COUNT
  - 8.4 END OF PASS MESSAGES
  - 8.5 ITERATIONS
  - 8.6 T BIT TRAPPING
  - 8.7 ACT11 COMPATABILITY
  - 8.8 PSW TABLE
  - 8.9 I/O DEVICE ADDRESS MODIFCATIONS
  - 8.10 POWER FAILURE
- 9.0 PROGRAM DESCRIPTION
  - 9.1 UNIBUS EXERCISER FUNCTION
  - 9.2 MASS BUS TESTER FUNCTION
  - 9.3 LINE CLOCK INITIALIZATION
  - 9.4 RELOCATION ALGORITHM



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  
145  
146  
147  
148  
149  
150  
151  
152  
153

1.0 ABSTRACT

-----  
THIS PROGRAM IS DESIGNED TO BE A COMPREHENSIVE CHECK OF THE PDP-11/6X SERIES CPU CLUSTER. THE PROGRAM EXECUTES EACH INSTRUCTION IN ALL ADDRESS MODES AND INCLUDES TESTS FOR TRAPS, INTERRUPTS, FLOATING POINT, MEMORY MANAGEMENT, MEMORY, THE UNIBUS, AND THE MASS BUS. IF NOT DESELECTED, THE PROGRAM RELOCATES THE TEST CODE THROUGHOUT MEMORY (0-124K). ALSO, IF NOT DESELECTED, THE PROGRAM WILL RELOCATE USING AVAILABLE DISKS (RPO3, RK05, RPO4, RS03/4). SEE SECTION 9.4 FOR A DESCRIPTION OF RELOCATION.

SINCE WORST CASE TESTING OCCURS WITH ALL SWITCHES DOWN, PRECAUTIONS MUST BE TAKEN TO ENSURE THE PROTECTION OF USER DISKS. REFER TO SECTION 7.0 FOR A DESCRIPTION OF WARNINGS AND EXCEPTIONS.

2.0 REQUIREMENTS

2.1 EQUIPMENT

-----  
PDP-11/6X SERIES CPU WITH 16K OF MEMORY, A LINE CLOCK, AND AN LA30 (OR EQUIVALENT) CONSOLE TERMINAL.

2.1.1 OPTIONAL EQUIPMENT USED

- 1. UNIBUS EXERCISER
- 2. MASS BUS TESTER
- 3. RP11/RPO3, RK11/RK05, RH11/RPO4, RH11/RS03/RS04

2.2 STORAGE

-----  
THE PROGRAM LOADS INTO THE FIRST 12K OF MEMORY AND RUNS IN ALL MEMORY (EXCLUSIVE OF THE XXDP MONITOR IF RUNNING IN CHAIN MODE).

2.3 PRELIMINARY PROGRAMS

-----  
ALTHOUGH THIS PROGRAM IS A TEST OF THE CPU, IT IS ADVISABLE THAT THE CPU (AND FLOATING POINT) DIAGNOSTICS RUN FIRST. THESE CONSIST OF:

- DQKDA DQFPA
- DQKDB DQFPB
- DQKKA DQFPC
- DQKTA DQFPD
- DQKUA



E01

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 4

154  
155

3.0 LOADING PROCEDURE



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  
1973.1 METHOD  
-----

THE PROGRAM IS SUPPLIED ON THE DIAGNOSTIC MEDIA. REFER TO THE XXDP OPERATING MANUAL FOR FURTHER INFORMATION. THE PROGRAM CAN ALSO BE DIRECTLY LOADED USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

IF LOADING A BINARY PAPER TAPE, BE SURE THE SWITCH REGISTER IS CLEARED AFTER THE ABSOLUTE LOADER IS LOADED.

4.0 STARTING PROCEDURE  
-----4.1 CONSOLE SWITCH SETTINGS  
-----

CHECK THAT THE SWITCH REGISTER WAS ZERO IF THE PROGRAM WAS LOADED FROM PAPER TAPE.

SEE SECTION 5.1

4.2 STARTING ADDRESSES  
-----

THE STARTING ADDRESS FOR THE EXERCISER IS 200.

4.3 PROGRAM AND OPERATOR ACTION  
-----

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3)
2. CHECK FOR ANY SYSTEM DISK PACKS OR CONFIGURATION EXCEPTIONS AS DESCRIBED IN SECTION 7.0.
3. LOAD ADDRESS 200
4. SET SWITCHES (SEE SECTION 5.1)
5. PRESS START
6. THE PROGRAM WILL LOOP AND MESSAGES WILL BE TYPED AT THE END OF EACH SUB-PASS AND EACH PASS. (SEE SECTION 8.4 FOR A DESCRIPTION OF THE MESSAGES)



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  
2525.0 OPERATING PROCEDURE  
-----5.1 OPERATIONAL SWITCH SETTINGS  
-----

SW15 (100000) HALT ON ERROR

THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED. AN ERROR MESSAGE IS TYPED AND THE PROCESSOR WILL HALT. PRESSING CONTINUE WILL RESUME TESTING.

SW14 (040000) LOOP ON TEST

THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.

SW13 (020000) INHIBIT ERROR  
TYPEOUT

THIS SWITCH WHEN SET INHIBITS THE ERROR TYPEOUT.

SW12 (010000) INHIBIT UBE

THIS SWITCH WHEN SET INHIBITS THE INITIALIZATION OF THE UNIBUS EXERCISER. SEE SECTION 9.1 FOR A DESCRIPTION OF THE UBE FUNCTION.

SW11 (004000) INHIBIT SUB-  
TEST ITERATION

THIS SWITCH WHEN SET INHIBITS SUBTEST ITERATION AFTER THE FIRST PASS. EACH SUBTEST IS EXECUTED 10 TIMES BEFORE THE NEXT SUBTEST IS RUN. SETTING SW11 CAUSES EACH TEST TO BE EXECUTED ONCE BEFORE STARTING THE NEXT SUBTEST.

SW10 (002000) RING BELL  
ON ERROR

THIS SWITCH WHEN SET WILL RING THE BELL WHEN AN ERROR IS DETECTED.

SW9 (001000) LOOP ON ERROR

THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE EVEN IF THE FAILURE IS INTERMITTANT. SEE SECTION 6.1 FOR A DESCRIPTION OF LOOPING ON RELOCATION ERRORS.

SW8 (000400) RELOCATE WITH  
CPU ONLY

THIS SWITCH WHEN SET WILL CAUSE RELOCATION TO BE DONE BY THE CPU INSTEAD OF A DISK. SEE SECTION 9.4 FOR A DESCRIPTION OF RELOCATION.



253  
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

SW7 (000200) INHIBIT SYSTEM  
SIZE TYPEOUT

THIS SWITCH WHEN SET WILL INHIBIT THE TYPEOUT OF THE SWITCH DEFINITIONS AND THE DISKS THAT WILL BE USED FOR RELOCATION. (TYPEOUT ONLY OCCURS WHEN THE PROGRAM IS DUMPED)

SW6 (000100) INHIBIT RELOCATION

THIS SWITCH WHEN SET WILL INHIBIT ALL RELOCATION. DO NOT CHANGE THIS SWITCH WHILE THE PROGRAM IS RUNNING.

SW5 (000040) INHIBIT ROUND  
ROBIN

THIS SWITCH WHEN SET WILL ONLY RELOCATE USING THE DEVICE SELECTED BY SWITCHES <2:0> RATHER THAN ALL AVAILABLE DEVICES.

SW4 (000020) INHIBIT RANDOM  
DISK ADDRESS

THIS SWITCH WHEN SET WILL CAUSE RELOCATION TO ALWAYS START AT ADDRESS 0 ON THE DISK(S).

SW3 (000010) INHIBIT MBT

THIS SWITCH WHEN SET INHIBITS THE INITIALIZATION OF THE MASS BUS TESTER. SEE SECTION 9.2 FOR A DESCRIPTION OF THE MBT FUNCTION.

SW2-SW0 (0 - 7) DEVICE CODES

THESE SWITCHES (ALONG WITH SW5) CAUSE THE PROGRAM TO RELOCATE THE TEST CODE USING THE DEVICE SPECIFIED BELOW:

VALUE	DEVICE
0	RP11/RP03
1	RK11/RK05
2	NOT USED
3	NOT USED
4	RH11/RP04
5	RH11/RS03/RS04
6	NOT USED
7	NOT USED

NOTE

WHEN RELOCATING VIA A SPECIFIC DEVICE, SET IN THE VALUE(SW<2:0>) TO SELECT THE DEVICE THEN SET SWITCH 5.



I01

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 8

309  
310

UNIT 0 OF THE LOAD DEVICE IS MARKED NOT  
PRESENT IF PROGRAM WAS LOADED IN CHAIN



311  
312  
313  
314  
315  
316  
317  
318  
319  
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

MODE (XXDP), AND THEREFORE WILL NOT BE  
USED TO RELOCATE.

5.2 DISPLAY REGISTER

WHILE THE PROGRAM IS RUNNING, THE LOW BYTE OF THE DISPLAY REGISTER CONTAINS THE SUBTEST NUMBER AND THE HIGH BYTE CONTAINS BITS <11:4> OF KERNEL PAR0. THESE BITS, OF KERNEL PAR0, CORRESPOND TO BITS <17:10> OF THE PHYSICAL ADDRESS OF THE RELOCATED CODE. WHEN AN ERROR IS DETECTED AND LOOP ON ERROR IS SELECTED, THE HIGH BYTE CONTAINS THE ERROR COUNT.

5.3 OPERATOR ACTION

WHEN THE PROGRAM IS LOADED\* AND STARTED WITH SWITCH 7 EQUAL TO ZERO THE PROGRAM WILL TYPEOUT THE DISKS AND UNIT NUMBERS THAT WILL BE USED FOR RELOCATION AND THEN WAIT FOR THE OPERATOR TO TYPE A CHARACTER. THIS IS TO ALLOW THE OPERATOR TO WRITE PROTECT ANY DRIVE THAT IS NOT TO BE USED. IF THERE ARE NO DEVICES AVAILABLE FOR RELOCATION, OPERATOR ACTION IS NOT REQUIRED.

IF THE PROGRAM IS LOADED VIA ACT11 IN QV OR AA OR WITH XXDP IN CHAIN MODE NO OPERATOR ACTION IS REQUIRED AND ALL DISKS NOT WRITE PROTECTED (EXCEPT FOR THE XXDP MEDIA) WILL BE USED FOR RELOCATION.

\*EXCEPT CHAIN MODE, QV(MANUFACTURING ONLY), OR AUTO ACCEPT (MANUFACTURING ONLY)

6.0 ERRORS

6.1 ERROR HALTS AND DESCRIPTION

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE ERROR HANDLING ROUTINE (\$ERROR). IF HALT ON ERROR IS ENABLED, THE PROCESSOR WILL TYPE THE ERROR MESSAGE AND THEN HALT. PRESSING CONTINUE WILL CAUSE TESTING TO RESUME.

THERE ARE MANY DIFFERENT TYPES OF ERRORS. NO MATTER WHICH TYPE OCCURS A MINIMUM SET OF INFORMATION IS TYPED AS FOLLOWS:

HHH:MM:SS  
ERRORPC PHYSC PC PSW TEST NO SUBPAS-PASS CNT  
UUUUUU VVVVVV WWWWWW YYYYYY SSSSSS PPPPPP

WHERE:

UUUUUU = VIRTUAL PC OF THE ERROR CALL.



K01

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 10

367  
368

VVVVVV  
WWWWW

= PHYSICAL PC OF THE ERROR CALL.  
= PSW AT THE TIME OF THE ERROR CALL.



369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424

YYYYYY = TEST NUMBER.  
SSSSSS = SUB-PASS COUNT (0 THRU 3)  
PPPPPP = PASS COUNT

HHH:MM:SS REPRESENTS THE ELAPSED RUN TIME OF THE PROGRAM, SINCE THE MOST RECENT START, WHERE: HHH = HOURS, MM = MINUTES, AND SS = SECONDS.

THE VIRTUAL PC IS THE 16 BIT WORD THAT WAS PUSHED ON THE STACK WHEN THE ERROR CALL WAS MADE. THE PHYSICAL PC IS CALCULATED IN ONE OF TWO WAYS:

1. IF MEMORY MANAGEMENT IS OFF THE CONTENTS OF LOCATION "FACTOR" IS SUBTRACTED FROM THE VIRTUAL PC. THIS GENERATES THE CORRESPONDING PC FOR THE NON-RELOCATED CODE.
2. IF MEMORY MANAGEMENT IS ON THE CONTENTS OF THE APPROPRIATE PAR IS SHIFTED AND ADDED TO THE VIRTUAL PC TO GENERATE A PHYSICAL 18 BIT ADDRESS. IN THIS CASE THE VIRTUAL PC CORRESPONDS TO THE NON-RELOCATED CODE.

DEPENDING ON THE TYPE OF ERROR ADDITIONAL INFORMATION IS TYPED AS DESCRIBED BELOW.

6.1.1 UNEXPECTED TRAP TO 4

VIRTPC    PHYSPC    PSW    CPUERR  
VVVVVV    PPPPPP    YYYYYY    ZZZZZZ

VVVVVV = VIRTUAL PC THAT WAS PUSHED ON THE STACK WHEN THE TRAP OCCURRED.  
PPPPPP = PHYSICAL PC CALCULATED AS DESCRIBED ABOVE.  
YYYYYY = PSW THAT WAS PUSHED ON THE STACK.  
ZZZZZZ = CONTENTS OF THE CPU ERROR REGISTER(777766).

6.1.2 UNEXPECTED TRAP TO 114

VIRTPC    PHYSPC    PSW    MEM ERR REG  
VVVVVV    PPPPPP    YYYYYY    ZZZZZZ

V, P, AND Y = ARE THE SAME AS DESCRIBED IN 6.1.1.  
ZZZZZZ = CONTENTS OF THE MEMORY ERROR REGISTER (777744).

6.1.3 PARITY ERROR DURING DATA CHECK

THIS ERROR CAN ONLY OCCUR DURING THE DATA CHECK THAT IS MADE ON THE RELOCATED TEST CODE BEFORE IT IS EXECUTED. THIS CHECK IS MADE BY COMPARING THE UNRELOCATED CODE WITH THE RELOCATED CODE. THE SOURCE ADDRESS REFERS TO THE UNRELOCATED CODE AND THE DESTINATION ADDRESS TO THE RELOCATED CODE.

SRCADR    DSTADR    MEM ERR REG  
SSSSSS    DDDDDD    ZZZZZZ



MO1

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 12

425  
426

SSSSSS = VIRTUAL ADDRESS OF THE SOURCE DATA.



427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482

DDDDDD = PHYSICAL ADDRESS OF THE DESTINATION DATA.  
ZZZZZZ = CONTENTS OF MEMORY ERROR REGISTER (777744).

6.1.4 ERROR DURING DATA CHECK-RELOC WAS BY CP

THIS ERROR IS SIMILAR TO 6.1.3 EXCEPT INSTEAD OF A PARITY ERROR, IT IS A DATA COMPARISON ERROR. REFER TO SECTION 9.4.3 FOR A DESCRIPTION OF CP RELOCATION.

LOOP ON ERROR (SW<9>) HAS THE FOLLOWING EFFECT:

1. MEMORY MANAGEMENT OFF- IF SWITCH<9> IS SET, LOOPING WILL BE PERFORMED ON THE SECTION RELOCATION (SEE SECTION 9.4.1). IF SW<9> IS NOT SET, EXECUTION WILL CONTINUE AT THE BEGINNING OF THE NEXT SECTION.
2. MEMORY MANAGEMENT ON- IF SW<9> IS SET, LOOPING WILL BE PERFORMED ON THE PROGRAM RELOCATION (SEE SECTION 9.4.2) TO THE SAME MEMORY SPACE THAT FAILED. IF SW<9> IS NOT SET, PROGRAM RELOCATION WILL BE RETRIED IN THE SAME MEMORY SPACE.

6.1.5 ERROR DURING DATA CHECK-RELOC WAS BY I/O

THIS ERROR IS THE SAME AS 6.1.4 EXCEPT RELOCATION WAS PERFORMED VIA A DISK RATHER THAN THE CP. THE ERROR PRINTOUT WILL IDENTIFY WHICH DEVICE AND DRIVE NUMBER TRANSFERRED THE PARTICULAR WORD THAT FAILED. REFER TO SECTION 9.4.4 FOR A DESCRIPTION OF I/O RELOCATION.

LOOP ON ERROR (SW<9>) HAS THE FOLLOWING EFFECT:

1. IF SW<9> IS SET, THE DEVICE THAT RELOCATED THE WORD (THAT CAUSED THE DATA CHECK ERROR) IS INITIATED TO DO THE SAME TRANSFER WITH THE SAME DISK ADDRESS AND MEMORY ADDRESSES. THIS TRANSFER WILL CONTINUALLY BE INITIATED AND CHECKED UNTIL SW<9> IS NOT SET.

6.1.6 DEVICE ERROR

THIS ERROR OCCURS IF A DEVICE ERROR OCCURS WHILE THE DEVICE IS DOING A TRANSFER. THE DEVICE AND DRIVE NUMBER ARE IDENTIFIED AND THE CONTENTS OF THE DEVICE REGISTERS ARE TYPED.

WHEN SW<9> (LOOP ON ERROR) IS SET, THE DEVICE THAT FAILED IS CONTINUALLY RESTARTED WITH THE SAME DISK ADDRESS, MEMORY ADDRESS, AND FUNCTION THAT CAUSED THE ERROR.

IF SW<9> IS NOT SET, RELOCATION IS RESTARTED.

6.1.7 UNIBUS EXERCISER FAILED

CC	BUSADR	CR2	CR1	PHYS BUS ADR
XXXXXX	VVVVVV	WWWWW	YYYYYY	ZZZZZZ



802

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 14

483  
484

XXXXXX  
VVVVVV

= CYCLE COUNT.  
= VIRTUAL BUS ADDRESS THAT THE UBE FAILED AT

485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540

WWWWW = CONTROL REGISTER NUMBER 2  
YYYYYY = CONTROL REGISTER NUMBER 1  
ZZZZZZ = PHYSICAL MEMORY ADDRESS THAT THE UBE FAILED AT

THE PHYSICAL MEMORY ADDRESS IS CALCULATED BY ADDING THE APPROPRIATE MAP REGISTER TO THE VIRTUAL BUS ADDRESS, FORMING A REAL 18 BIT MEMORY ADDRESS.

6.1.8 UBE NON-EXISTANT MEMORY ERROR

THIS ERROR ONLY OCCURS WHEN THE "NO SLAVE SYNC" ERROR OCCURS IN THE UNIBUS EXERCISER. ONLY THE PHYSICAL ADDRESS THAT TIMED OUT IS TYPED. THIS ERROR MIGHT INDICATE THAT THERE IS A HOLE IN MEMORY.

6.1.9 MASS BUS TESTER FAILED

CS1 WRDCNT BUSADR BADREX MR2 CS2 ST  
AAAAA BBBB CCCCC DDDDD EEEEE FFFFF GGGGG

ER CS3  
HHHHH JJJJJ

AAAAA = CONTROL AND STATUS REGISTER #1 (760100).  
BBBBB = WORD COUNT REGISTER (760102).  
CCCCC = BUS ADDRESS REGISTER (760104).  
DDDDD = BUS ADDRESS EXTENDED REGISTER (760174).  
EEEEEE = MAINTENANCE REGISTER #2 (760106).  
FFFFFF = CONTROL AND STATUS REGISTER #2 (760110).  
GGGGG = STATUS REGISTER (760112).  
HHHHH = ERROR REGISTER (760114).  
JJJJJ = CONTROL AND STATUS REGISTER #3 (760176).

6.1.10 MBT NON-EXISTANT MEMORY ERROR

THIS IS THE SAME AS 6.1.8 EXCEPT THAT IT IS DETECTED BY THE NEXM BIT IN CS2 OF THE MBT.

6.1.11 FLOATING POINT ERROR

THIS ERROR WILL ONLY OCCUR IF THE LEFT AND RIGHT HAND SIDES OF THE FLOATING POINT IDENTITIES DO NOT AGREE WITHIN THE EXPECTED TOLERANCE. THE VALUE OF THE CALCULATIONS ARE TYPED OUT.

THIS ERROR SHOULD ONLY BE A FUNCTION OF THE FLOATING POINT PROCESSOR AND THE FPP DIAGNOSTICS (DQFPA-DQFPD) SHOULD BE USED TO ISOLATE THE PROBLEM.

6.1.12 DEVICE HUNG

THIS ERROR WILL OCCUR IF A DEVICE DOES NOT FINISH ITS RELOCATION FUNCTION WITHIN 2 SECONDS AFTER ITS INITIATION. REFER TO SECTION 9.4.4.4 TO DETERMINE WHICH DEVICE AND DRIVE



002

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 16

541

IS HUNG.

542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
5976.2 ERROR RECOVERY  
-----

DIFFERENT TYPES OF ERRORS RECOVER IN DIFFERENT WAYS AS DESCRIBED BELOW.

## 6.2.1 ERRORS WITHIN SUBTESTS

EXECUTION STARTS WITH THE INSTRUCTION FOLLOWING THE ERROR CALL.

## 6.2.2 RELOCATION WITH MEMORY MGMT. OFF

EXECUTION STARTS AT THE BEGINNING OF THE NEXT SECTION.

## 6.2.3 DEVICE ERROR OR CP RELOCATION WITH MEMORY MGMT. ON

RELOCATION IS RESTARTED AT THE BEGINNING OF WHATEVER TYPE OF RELOCATION THE ERROR WAS FOUND IN.

## 6.2.4 UNEXPECTED TRAPS (4,10,114,250)

EXECUTION STARTS AT THE ADDRESS POINTED TO BY LOCATION "SLPERR". THIS LOCATION CONTAINS THE ADDRESS+2 OF THE MOST RECENTLY EXECUTED "SCOPE" INSTRUCTION. HOWEVER, IF A SECOND TRAP OCCURS BEFORE THE FIRST IS COMPLETELY SERVICED, THE PROGRAM WILL HALT.

7.0 WARNINGS AND EXCEPTIONS  
-----7.1 WARNINGS  
-----

ANY DRIVE THAT IS NOT "WRITE PROTECTED" WILL BE WRITTEN ON (EXCEPT UNIT 0 OF THE XXDP LOAD DEVICE IN CHAIN MODE).

WHEN THE PROGRAM IS DUMPED (SEE SECTION 5.3) AND SW<7> IS SET, THE DEVICES AND DRIVES THAT ARE NOT WRITE PROTECTED WILL BE IDENTIFIED ON THE TERMINAL. BEFORE TYPING A CHARACTER TO CONTINUE, A DRIVE CAN BE WRITE PROTECTED WITHOUT CAUSING AN ERROR BECAUSE, THE SYSTEM IS SIZED AGAIN.

7.2 EXCEPTIONS  
-----

IF ANY OF THE DEVICES IS LOCATED AT A NON-STANDARD ADDRESS (SEE BELOW), THE DEVICE REGISTER ADDRESS TABLES (IN "COMMON TAGS") SHOULD BE CHANGED TO THE CORRECT ADDRESSES. FOLLOWING IS THE DEFAULT ADDRESS OF THE CONTROL AND STATUS REGISTER OF EACH DEVICE:



F02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 18

598  
599

RK05----177404  
RP04----176700

600  
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

## RS03/4--172040

IF THE SYSTEM HAS BOTH AN RPO3 AND AN RPO4, THE BRANCH INSTRUCTION AT 1005, IN THE "SIZE ROUTINE" MUST BE REPLACED BY A NOP (240) FOR BOTH DEVICES TO BE USED. THIS BRANCH IS APPROXIMATELY AT ADDRESS 4706. ALSO, THE OPERATER MUST CHANGE THE RPO4 ADDRESSES IN THE TABLE IN THE "COMMON TAGS" AREA.

8.0 MISCELLANEOUS  
-----8.1 EXECUTION TIME  
-----

THE EXECUTION TIME IS DEPENDENT ON THE AMOUNT OF MEMORY ON THE SYSTEM. FOLLOWING ARE TWO TYPICAL RUN TIMES:

1. MANUFACTURING BASIC LINE-16K MEMORY, UBE, MBT, AND NO DISKS---3 MINUTES.
2. SYSTEM-128K MEMORY, 2 RK05'S, RPO4, AND 2 RS04'S --15 MINUTES.

8.2 STACK POINTER  
-----

THE STACK POINTER IS SET TO 700.

## NOTE

WHEN THE PROGRAM IS RUNNING IN USER MODE, THE USER STACK POINTER IS SET TO 700 AND THE KERNEL STACK POINTER IS SET TO 1200. THE KERNEL STACK POINTER IS USED ONLY FOR THE ERROR AND INTERRUPT SERVICE ROUTINES.

8.3 PASS COUNT  
-----

THERE ARE TWO WORDS USED FOR EFFECTIVE PASS COUNT. LOCATION "SUBPASS" AND "SPASS". SUBPASS CONTAINS THE ASCII REPRESENTATION OF THE SUBPASS COUNT. THIS IS USED TO INDEX THE PSW TABLE (SEE SECTION 8.8).

FOUR SUBPASSES ARE EXECUTED FOR EACH PASS. THIS ALLOWS ALL PSW COMBINATIONS TO BE TESTED BEFORE REPORTING END OF PASS.

8.4 END OF PASS MESSAGES  
-----

AT THE END OF EACH SUBPASS THE SUBPASS NUMBER (THAT IS BEING STARTED) IS TYPED FOLLOWED BY "THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789". IF RUNNING ON ACT11 QV OR AA,



H02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 20

656  
657  
658

ONLY THE SUB-PASS NUMBER IS TYPED. AT THE END OF EACH PASS  
THE ELAPSED RUN TIME AND THE MESSAGE "END PASS X TOTAL ERRORS  
SINCE LAST REPORT Y" IS TYPED. A SAMPLE OF WHAT A COMPLETE

659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714

PASS LOOKS LIKE IS SHOWN BELOW:

0THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789  
1THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789  
2THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789  
3THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789  
END PASS 1 TOTAL ERRORS SINCE LAST REPORT 0

#### 8.5 ITERATIONS

-----

SUB-TEST ITERATIONS ARE NOT PERFORMED UNTIL THE PASS COUNT (\$PASS) IS NON-ZERO. THIS MAKES A QV PASS AS SHORT AS POSSIBLE.

AFTER THE FIRST PASS, FULL 10 OCTAL ITERATIONS ARE PERFORMED ON EACH SUBTEST.

#### 8.6 T-BIT TRAPPING

-----

T BIT TRAPPING IS CONTROLLED BY THE PSW TABLE. THE DEFAULT CONDITION IS TO RUN WITH THE T-BIT ON DURING SUBPASSES 2, 4, AND 6.

#### 8.7 ACT-11 COMPATABILITY

-----

THE PROGRAM IS FULLY ACT-11 COMPATABLE. THE PROGRAM OPERATES IN THE ACT-11 MODE UNDER APT.

#### 8.8 PSW TABLE

-----

AT THE END OF THE PROGRAM, JUST BEFORE THE MESSAGES, IS THE PSW TABLE. THIS TABLE CONTROLS WHAT MODE WILL BE EXECUTED ON A SUBPASS. REFER TO SECTION 9.4.2 FOR A DESCRIPTION OF HOW THIS TABLE IS USED BY THE PROGRAM. THIS TABLE MAY BE MODIFIED IF DESIRED.

#### 8.9 I/O DEVICE ADDRESS MODIFICATION

-----

TO MODIFY THE PROGRAM ADDRESS OF THE I/O DEVICES PATCH THE APPROPRIATE DEVICE TABLE (IN THE COMMON TAGS AREA) TO THE DESIRED ADDRESSES.

IF YOU ARE PATCHING THE RPO3 OR RPO4 SEE SECTION 7.2.

#### 8.10 POWER FAIL

-----

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP),, THE WORD



J02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 22

715

"POWER" IS TYPED ON THE TERMINAL AND THE PROGRAM RESTARTS.

716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
7719.0 PROGRAM DESCRIPTION  
-----

THE PROGRAM IS DIVIDED INTO 9 SECTIONS OF POSITION INDEPENDENT RELOCATABLE TEST CODE. EACH SECTION IS APPROXIMATELY 1K WORDS LONG.

WHEN THE PROGRAM IS INITIALLY LOADED AND STARTED IT WILL IDENTIFY ITSELF AND TYPE THE FUNCTION OF THE SWITCH REGISTER AND THE DEVICES AND DRIVES THAT WILL BE USED FOR RELOCATION, IF SW7=0. IT WILL ALSO TYPE THE CP OPTIONS AVAILABLE INDICATOR WORD (OPT.CP). THE CONTENTS OF OPT.CP CONTAIN THE FOLLOWING INDICATORS:

BIT15	=	NOT USED
BIT14	=	NOT USED
BIT13	=	NOT USED
BIT12	=	NOT USED
BIT11	=	NOT USED
BIT10=1/0	=	MBT AVAILABLE/NOT AVAILABLE
BIT09=1/0	=	KW11-L AVAILABLE/NOT AVAILABLE
BIT08=1/0	=	CONSOLE TTY AVAILABLE/NOT AVAILABLE
BIT07=1/0	=	UBE AVAILABLE/NOT AVAILABLE
BIT06=00	=	NOT USED

FOLLOWING IS A BRIEF DESCRIPTION OF EACH SECTION:

SECTION 0 THIS SECTION CAUSES A 256 WORD 3 XOR 9 TEST PATTERN TO BE RELOCATED THROUGHOUT MEMORY 0 - 28K.

NOTE: THIS SHOULD NOT BE CONSTRUED TO BE A COMPLETE MEMORY TEST.

SECTION 1 THIS SECTION TESTS THE UNARY INSTRUCTION SET EXECUTING EACH UNARY INSTRUCTION IN EACH ADDRESS MODE (EXCLUDING UNARY INSTRUCTIONS USING ADDRESS MODE 7).

SECTION 2 THIS SECTION TESTS THE UNARY INSTRUCTIONS USING ADDRESS MODE 7 AND BINARIES IN ALL ADDRESS MODES (EXCLUDING BINARY BYTE OPS USING ADDRESS MODE 7).

SECTION 3 THIS SECTION TESTS BINARY BYTE OPS USING ADDRESS MODE 7, JMP, JSR AND PROGRAM TRAP (IOT, TRAP, AND EMT) INSTRUCTION.

SECTION 4 THIS SECTION CHECKS THAT EACH BIT IN THE PROCESSOR STATUS WORD (PSW) CAN BE SET CLEARED, RESERVED INSTRUCTIONS, AND ODD ADDRESS TRAPS. NOTE THAT BITS 12 AND 14 IN THE PSW ARE COPIES OF BITS 13 AND 15 RESPECTIVELY. THE WCS IS TURNED OFF AT THE BEGINNING OF THE PROGRAM BY INITIALIZING THE WHAMI REGISTER SO THAT THE RESERVED INSTRUCTIONS TRAP.



L02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 24

772  
773

SECTION 5 THIS SECTION CHECKS THE SXT, XOR, SOB, MARK, RTT

774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829

AND RTT INSTRUCTIONS.

SECTION 6 THIS SECTION CHECKS THE ASH, ASHC, MUL, DIV INSTRUCTIONS. ALSO CHECKED IN THIS SECTION ARE THE MED INSTRUCTION AND ERROR LOGGING CAPABILITIES.

SECTION 7 THIS SECTION CHECKS THE STACK LIMIT REGISTER MEMORY MANAGEMENT ABORT LOGIC AND THE MEMORY MANAGEMENT REGISTERS.

SECTION 8 THIS SECTION CHECKS THE FLOATING POINT PROCESSOR.

FOLLOWING SECTION 8 ARE TWO ROUTINES TO CHECK THE TELETYPE PRINTER LOGIC AND A ROUTINE TO START THE KW11-L CLOCK. IF THE KW11-L IS AVAILABLE THE PRIORITY ARBITRATION LOGIC IS TESTED.

9.1 UNIBUS EXERCISER(UBE)  
-----

ANY ONE OF 4 UBE'S WILL BE USED. THE PROGRAM LOOKS FOR A UBE AT ADDRESSES 770000, 770020, 770040, AND 770060.

TEST 106 WILL INITIATE THE UNIBUS EXERCISER IF IT IS PRESENT. THIS IS ONLY DONE ON PASS 1 - SUBPASS 1. SINCE FROM THAT POINT ON, THE SERVICE ROUTINE TAKES CARE OF RESTARTING IT.

THE UBE IS INITIALLY SET UP WITH A BUS ADDRESS OF 0. THE FUNCTION THAT IS LOADED IS "DATA IN PAUSE-DATA OUT BYTE". THE WORD COUNT IS SET FOR 4K BYTES. IT IS ALSO SET TO INTERRUPT ON LEVEL 5.

WHEN AN INTERRUPT OCCURS A CHECK IS MADE TO SEE IF IT WAS CAUSED BY AN ERROR. IF THERE WAS NO ERROR, 776 IS LOADED AS THE BUS ADDRESS AND THE UBE IS STARTED AGAIN. ON THE NEXT INTERRUPT 1774 (776+776) IS LOADED AS THE BUS ADDRESS. THIS SEQUENCE CONTINUES UNTIL A MEMORY TIMEOUT ERROR OCCURS.

WHEN AN ERROR OCCURS A CHECK IS MADE TO SEE IF IT WAS CAUSED BY A MEMORY TIMEOUT. IF IT WAS, THE ADDRESS IN THE UBE BUS ADDRESS REGISTER IS COMPARED WITH THE LAST ADDRESS IN THE SYSTEM (MXMMHI AND MXMML0). IF THEY ARE THE SAME (NO HOLES IN MEMORY) THE UBE IS RESTARTED AT ADDRESS 0 AND THE ABOVE SEQUENCE IS REPEATED. IF THE ADDRESSES ARE NOT THE SAME A MEMORY-HOLE ERROR IS REPORTED.

IF THE ERROR WAS NOT DUE TO A TIMEOUT A UBE ERROR IS REPORTED.

9.2 MASS BUS TESTER(MBT)  
-----

ANY ONE OF 4 MBT'S WILL BE USED. THE PROGRAM LOOKS FOR AN MBT AT ADDRESSES 770100, 770200, 770300, AND 770400. IF AN MBT IS



N02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 26

830  
831

FOUND THE DRIVE TYPE REGISTER (770X26) IS CHECKED TO MAKE  
SURE THAT IT REALLY IS AN MBT.

832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887

TEST 106 ALSO INITIATES THE MASS BUS TESTER. AGAIN, THIS IS ONLY DONE ON PASS 1 - SUBPASS 1 SINCE THE SERVICE ROUTINE KEEPS IT RUNNING.

THE BUS ADDRESS REGISTER IS INITIALLY SET TO 0, THE WORD COUNT TO 2K WORDS, AND A READ FUNCTION IS INITIATED.

WHEN AN INTERRUPT OCCURS AN ERROR CHECK IS MADE. THIS ERROR CHECK IS THE SAME AS THAT DESCRIBED FOR THE UBE. IF THERE WAS NO ERROR, THE WORD COUNT IS RELOADED AND THE FUNCTION IS ISSUED. THE BUS ADDRESS REGISTER IS NOT CHANGED SO IT WILL CONTINUE FROM WHERE IT LEFT OFF.

### 9.3 LINE CLOCK INITIALIZATION

-----

TEST 106 TURNS ON THE LINE CLOCK. TWO LOCATIONS IN "COMMON TAGS" KEEP TRACK OF THE ELAPSED RUN TIME OF THE PROGRAM. WHEN THE CLOCK INTERRUPTS, THE LOW BYTE OF LOCATION "LTICKS" IS INCREMENTED. WHEN THIS BYTE GETS TO 60(DECIMAL) IT IS CLEARED AND THE HIGH BYTE IS INCREMENTED(SECONDS). WHEN THE SECOND COUNT GETS TO 60(DECIMAL) LOCATION "MTICKS" IS INCREMENTED AND LTICKS IS CLEARED. THIS GIVES THE TIMER A 64K DECIMAL MINUTE RANGE.

#### NOTE

FOR THE UBE, MBT, AND LINE CLOCK, WHEN AN INTERRUPT OCCURS, PROGRAM EXECUTION RETURNS TO KERNEL MODE AND THE KERNEL PAR'S ARE MAPPED DOWN TO THE 0-12K BANK OF MEMORY. UPON RETURNING FROM THE INTERRUPT THE PAR'S ARE MAPPED BACK TO WHERE THEY WERE AND THE PREVIOUS PROCESSOR MODE IS RESTORED.

### 9.4 RELOCATION ALGORITHM

-----

#### 9.4.1 SECTION RELOCATION

AS EACH SECTION IS ENTERED THE VIRTUAL START ADDRESS IS SAVED IN LOCATION "FRSTAD" AND THE RELOCATION FACTOR (BYTE OFFSET FROM NON-RELOCATED CODE) IS CALCULATED AND SAVED IN LOCATION "FACTOR". THE TEST CODE IS THEN EXECUTED.

AT THE END OF EACH SECTION, CONTROL IS TRANSFERRED TO THE "RELOCATION ROUTINE". IF SW<8> IS SET, THIS ROUTINE WILL RELOCATE THE SECTION VIA THE CP (SEE 9.4.3). IF SW<8> IS NOT SET, THE LENGTH OF THE SECTION IS CALCULATED, SAVED AS A WORD COUNT, AND CONTROL IS TRANSFERRED TO THE "I/O MONITOR" (SEE SECTION 9.4.4) WHICH RELOCATES THE SECTION BY USING A DISK.



C03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 28

888  
889

EACH SECTION IS INITIALLY RELOCATED TO THE END ADDRESS OF THE  
PROGRAM. SUBSEQUENT RELOCATIONS START AT THE END OF THE

890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945

PREVIOUS RELOCATION. FOR EXAMPLE: IF SECTION 0 IS 1000 BYTES LONG AND THE END ADDRESS OF THE PROGRAM IS 60000, THE FIRST RELOCATION STARTS AT ADDRESS 60000, THE SECOND AT 61000, THE THIRD AT 62000, ETC. THIS CONTINUES UNTIL 28K OR THE END OF MEMORY HAS BEEN REACHED AT WHICH TIME EXECUTION GOES TO THE START OF THE NEXT SECTION AND THE PROCESS REPEATS WITH THE NEW SECTION.

EACH SECTION IS WRITTEN IN POSITION INDEPENDENT CODE SO THAT IT CAN BE RELOCATED AND EXECUTED WITHOUT THE USE OF MEMORY MANAGEMENT.

#### 9.4.2 PROGRAM RELOCATION

WHEN ALL NINE SECTIONS HAVE BEEN RELOCATED AND EXECUTED THRU 28K (SEE SECTION 9.4.1), MEMORY MANAGEMENT IS SETUP ACCORDING TO THE VALUE IN LOCATION "NEXPAR" (MEMORY MANAGEMENT WILL NOT BE TURNED ON IF THERE IS LESS THAN 28K OF MEMORY). THIS VALUE IS INITIALIZED TO 600 (OR 1600 IF RUNNING UNDER THE XXDP MONITOR), MAKING RELOCATION START AT ADDRESS 60000 (OR 160000). THE "I/O MONITOR" IS THEN ENTERED (SEE SECTION 9.4.4) TO RELOCATE THE PROGRAM. WHEN THE I/O MONITOR COMPLETES THE RELOCATION, EXECUTION IS TRANSFERED TO THE START OF THE PROGRAM AT THE RELOCATED POSITION.

EACH SECTION IS EXECUTED ONLY ONCE WITH MEMORY MANAGEMENT ON. AT THE END OF SECTION 8, 77 IS ADDED TO "NEXPAR" AND RELOCATION IS PERFORMED AGAIN. THIS CAUSES THE NEXT RELOCATION TO MOVE UP BY 7700 BYTES. FOR EXAMPLE: IF NEXPAR=1600 THE FIRST RELOCATION STARTS AT ADDRESS 160000, THE SECOND AT ADDRESS 167700, THE THIRD AT 177600, ETC.

THIS CONTINUES UNTIL THE END OF MEMORY IS REACHED AND CONSTITUTES A SUB-PASS. THE PSW IS THEN SETUP FOR THE NEXT SUB-PASS AND THE PROGRAM RESTARTS.

THE VALUE FOR THE PSW IS TAKEN FROM THE TABLE (SEE SECTION 8.8). THE PARTICULAR ENTRY THAT IS USED IS OBTAINED BY INDEXING THE TABLE BY THE SUB-PASS NUMBER (SEE SECTION 8.3). FOR EXAMPLE, SUB-PASS 3 USES WORD 3 (THE FIRST WORD IS COUNTED AS ZERO) OF EACH TABLE. THEREFORE, TO CHANGE THE VALUE IN THE PSW ONLY REQUIRES CHANGING THE VALUE IN THE TABLE.

THE COMPLETION OF 4 SUB-PASSES CONSTITUTES A PASS AND AN END OF PASS MESSAGE IS TYPED. THE PROGRAM THEN RESTARTS IN PASS 2, SUB-PASS 0.

#### 9.4.3 RELOCATION VIA CP

IF SW<8> IS SET, BOTH SECTION AND PROGRAM RELOCATION (SEE SECTIONS 9.4.1 AND 9.4.2), ARE PERFORMED BY AN INSTRUCTION MOVE LOOP RATHER THAN A DISK. FOR EXAMPLE:



E03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 30

946  
947

1S: MOV (R0)+,(R2)+  
CMP R0,R3

948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003

BNE IS

WHERE R0 IS THE ADDRESS OF THE CODE BEING MOVED, R2 IS THE ADDRESS THAT IT IS BEING MOVED TO, AND R3 IS THE LAST ADDRESS THAT IS TO BE MOVED.

WHEN THIS IS FINISHED, THE RELOCATED DATA IS CHECKED BY AN INSTRUCTION COMPARE LOOP TO ENSURE THAT THE RELOCATION WAS PERFORMED CORRECTLY.

9.4.4 RELOCATION VIA I/O

IF SW<8> IS NOT SET, BOTH SECTION AND PROGRAM RELOCATION (SEE SECTION 9.4.1 AND 9.4.2), ARE PERFORMED BY WRITING THE DATA TO A DISK AND READING IT BACK TO THE RELOCATED POSITION. THIS RELOCATION IS CONTROLLED BY THE "I/O MONITOR".

9.4.4.1 SECTION RELOCATION

WHEN THE I/O MONITOR IS ENTERED FROM THE "RELOCATION ROUTINE" (SEE SECTION 9.4.1) A DEVICE IS SELECTED (SEE 9.4.4.3), THE MEMORY ADDRESSES (FROM AND TO) AND WORD COUNT ARE PASSED TO THE DEVICE HANDLER (SEE SECTION 9.4.4.4), AND THE HANDLER IS CALLED. WHEN THE HANDLER FINISHES, THE I/O MONITOR CHECKS THE RELOCATED DATA WITH AN INSTRUCTION COMPARE LOOP TO ENSURE THE RELOCATED DATA IS CORRECT, AND RETURNS TO THE "RELOCATION ROUTINE" (SEE 9.4.1).

9.4.4.2 PROGRAM RELOCATION

WHEN THE I/O MONITOR IS ENTERED FOR PROGRAM RELOCATION (SEE SECTION 9.4.2) THE BASE ADDRESS FOR THE RELOCATION IS CALCULATED FROM THE CONTENTS OF KERNEL PAR3 WHICH WAS SET UP WITH MEMORY MANAGEMENT (SEE 9.4.2). IF SW<8> IS SET, RELOCATION IS PERFORMED VIA THE CP (SEE SECTION 9.4.3).

IF SW<8> IS NOT SET, A DEVICE IS SELECTED (SEE 9.4.4.3), THE WORD COUNT IS SET TO 2K, AND THE MEMORY ADDRESSES (FROM AND TO) AND WORD COUNT ARE PASSED TO THE DEVICE HANDLER (SEE 9.4.4.4) AND THE HANDLER IS CALLED. THE I/O MONITOR THEN ADDS 2K TO THE MEMORY ADDRESSES, SELECTS ANOTHER DEVICE, PASSES THE ADDRESSES TO THE DEVICE HANDLER, AND CALLS THE HANDLER. THIS CONTINUES UNTIL ALL 12K HAS BEEN RELOCATED. THE RELOCATED DATA IS THEN CHECKED WITH AN INSTRUCTION COMPARE LOOP. THE RELOCATED PROGRAM IS THEN EXECUTED AS DESCRIBED IN 9.4.2.

9.4.4.3 DEVICE SELECTION

IF SW<5> IS NOT SET, AN INDEX IS PICKED UP FROM LOCATION "DEVINDX". THIS INDEX IS USED TO INDEX THE SYSTEM SIZE TABLE. THE SYSTEM SIZE TABLE CONSISTS OF 8 WORDS (ONE FOR EACH DEVICE TYPE). BITS <7:0> OF EACH WORD ARE USED TO INDICATE THE DRIVE



G03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 32

1004  
1005

NUMBERS THAT ARE AVAILABLE ON THE DEVICE, AND ARE INITIALIZED  
IN THE SIZE ROUTINE. BITS <15:8> OF EACH WORD ARE USED TO

1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061

INDICATE WHETHER THE DRIVE HAS BEEN USED FOR A DATA TRANSFER (UNIT USED BIT).

THE SYSTEM SIZE TABLE IS THEN SEARCHED, USING THE INDEX DESCRIBED ABOVE, FOR A DRIVE THAT HAS NOT BEEN USED. WHEN A DRIVE IS FOUND, THE "UNIT USED BIT" IS SET, THE CURRENT INDEX IS PUT BACK IN LOCATION DEVINDX, AND EXECUTION CONTINUES AS DESCRIBED IN 9.4.4.1 OR 9.4.4.2.

IF AN UNUSED UNIT IS NOT FOUND, ALL THE "UNIT USED" BITS ARE CLEARED AND THE SEARCH IS RESTARTED. IF THE SEARCH FINDS THE SYSTEM SIZE TABLE EMPTY (NO DEVICES ON THE SYSTEM), THE MESSAGE "NO I/O DEVICES" IS TYPED AND RELOCATION IS PERFORMED VIA THE CP AS DESCRIBED IN 9.4.3.

IF SW<5> IS SET, SW'S<2:0> ARE USED TO INDEX THE SYSTEM SIZE TABLE. IN THIS CASE ONLY ONE WORD OF THE TABLE IS USED CORRESPONDING TO THE DEVICE BEING SELECTED BY SW'S<2:0> (SEE SECTION 5.1). IN THIS MODE, A ROUND ROBIN SELECTION IS PERFORMED ON THE DRIVES OF THE SELECTED DEVICE.

#### 9.4.4.4 DEVICE HANDLERS

EACH DEVICE THAT IS USED FOR RELOCATION HAS A HANDLER. THESE HANDLERS ARE FUNCTIONALLY THE SAME.

THE HANDLER IS CALLED BY THE I/O MONITOR (SEE SECTION 9.4.4). IT FIRST CLEARS THE DONE BIT (BIT 7) IN THE HANDLER STATUS WORD. THIS PREVENTS THE MONITOR FROM CALLING THIS HANDLER AGAIN BEFORE IT IS FINISHED.

IF A "DEVICE HUNG" ERROR (SEE SECTION 6.1.12) IS DETECTED, THE HANDLER STATUS WORDS CAN BE EXAMINED TO DETERMINE WHICH DEVICE DID NOT FINISH (SET BIT 7). THE DRIVE CAN THEN BE DETERMINED BY LOOKING IN THE "DEVICE HANDLER UNIT NUMBER" TABLE. THE HANDLER STATUS WORDS AND DEVICE HANDLER UNIT NUMBER TABLES, ARE LOCATED IN THE "COMMON TAGS" AREA OF THE LISTING.

THEN THE HANDLER CALCULATES A DISK ADDRESS. THIS ADDRESS IS EITHER GENERATED FROM A RANDOM NUMBER (SW4=0) OR IS SET TO ZERO (SW4=1). THE DEVICE ID, UNIT NUMBER, AND CYLINDER ADDRESS ARE COMBINED AND PLACED IN THE "RUN TABLE" (RUNTBL). THE POSITION IN THE RUN TABLE CORRESPONDS TO WHICH 2K BLOCK OF THE PROGRAM IS BEING TRANSFERRED (I.E. THE FIRST 2K BLOCK IS IDENTIFIED BY WORD 1, THE SECOND 2K BY WORD 2, ETC.). THE BIT CONFIGURATION OF EACH WORD IN THE RUN TABLE IS AS FOLLOWS:

<15:13> = DEVICE ID  
<12:10> = UNIT NUMBER  
<9> = NOT USED  
<8:0> = CYLINDER ADDRESS

THE TRACK-SECTOR ADDRESS OF THE TRANSFER IS SAVED IN THE "RUN



I03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 34

1062  
1063

TRACK TABLE" (RUNTRAK). THE POSITION IN THIS TABLE IS AS  
DESCRIBED ABOVE. THE BIT CONFIGURATION OF EACH WORD IS THE

1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103

SAME AS THAT FOR THE DISK ADDRESS REGISTER FOR THE PARTICULAR DEVICE. BIT 15 IS USED TO INDICATE A DEVICE ERROR. IT IS SET BY THE DEVICE SERVICE ROUTINE. (SEE SECTION 9.4.4.5)

THE HANDLER THEN INITIALIZES THE DEVICE REGISTERS WITH ALL THE APPROPRIATE INFORMATION AND STARTS A WRITE FUNCTION. EXECUTION THEN RETURNS TO THE I/O MONITOR AT THE POINT WHERE THE HANDLER WAS CALLED.

#### 9.4.4.5 DEVICE SERVICE ROUTINES

EACH DEVICE THAT IS USED FOR RELOCATION HAS A SERVICE ROUTINE. THESE ROUTINES ARE ALL FUNCTIONALLY THE SAME.

THE ROUTINE IS ENTERED BY A DEVICE INTERRUPT. THE DEVICE IS CHECKED FOR ANY ERRORS. IF NO ERROR OCCURRED THE DEVICE REGISTERS ARE LOADED AND THE NEXT FUNCTION TO PERFORM IS INITIATED. THREE FUNCTIONS ARE EXECUTED: WRITE, WRITE CHECK, AND READ. ALL THE NECESSARY BUS ADDRESS INFORMATION IS CALCULATED BY THE I/O MONITOR, SO THE SERVICE ROUTINE JUST TAKES CARE OF THE DEVICE.

WHEN THE READ FUNCTION HAS BEEN COMPLETED SUCCESSFULLY, THE DONE BIT (BIT 7) IN THE HANDLER STATUS WORD IS SET.

UPON INITIATION OF A FUNCTION, OR COMPLETION OF ALL THREE FUNCTIONS, THE SERVICE ROUTINE RETURNS EXECUTION TO WHERE IT WAS WHEN IT WAS INTERRUPTED.

IF AN ERROR IS DETECTED, THE FUNCTION THAT FAILED IS RETRIED TWO MORE TIMES. IF THE ERROR IS STILL PRESENT THE DONE BIT AND THE ERROR BIT (BIT 15) IS SET IN THE HANDLER STATUS WORD ALONG WITH BIT 15, IN THE APPROPRIATE ENTRY, IN THE RUN TRACK TABLE, AND THE ROUTINE EXITS AS DESCRIBED ABOVE.



K03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 36

1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112

.TITLE MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
:\*COPYRIGHT (C) JANUARY, 1977  
:\*DIGITAL EQUIPMENT CORP.  
:\*MAYNARD, MASS. 01754  
:\*  
:\*  
:\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
:\*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
:\*

1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168

SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT USE
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	INHIBIT RELOCATION VIA I/O DEVICE
7	INHIBIT SYSTEM SIZE TYPEOUT
6	INHIBIT RELOCATION
5	INHIBIT ROUND ROBIN
4	INHIBIT RANDOM DISK ADDRESS
3	INHIBIT MBT
2	THESE THREE SWITCHES
1	ARE ENCODED TO SELECT RELOCATION
0	ON THE FOLLOWING DEVICES:
0...	RP11/RP03
1...	RK11/RK05
2...	NOT USED
3...	NOT USED
4...	RH11/RP04
5...	RH11/RS04
6...	NOT USED
7...	NOT USED

SWITCH	OCTAL VALUE
15	100/00
14	4/000
13	2/000
12	1/0000
11	4000
10	2000
9	1000
8	400
7	200
6	100
5	40
4	20
3	10
2	4
1	2

M03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER      MACY11 27(1006) 07-FEB-77 10:08 PAGE 38  
DQKDC.A.P11      07-FEB-77 09:58      OPERATIONAL SWITCH SETTINGS

1169

; \*            0            1



```

1170 .SBTTL BASIC DEFINITIONS
1171
1172 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
1173 STACK= 1200
1174 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
1175 .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL
1176
1177 ;*MISCELLANEOUS DEFINITIONS
1178 HT= 11 ;:CODE FOR HORIZONTAL TAB
1179 LF= 12 ;:CODE FOR LINE FEED
1180 CR= 15 ;:CODE FOR CARRIAGE RETURN
1181 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
1182 PS= 177776 ;:PROCESSOR STATUS WORD
1183 .EQUIV PS,PSW
1184 STKLM= 177774 ;:STACK LIMIT REGISTER
1185 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
1186 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
1187 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
1188
1189 ;*GENERAL PURPOSE REGISTER DEFINITIONS
1190 R0= %0 ;:GENERAL REGISTER
1191 R1= %1 ;:GENERAL REGISTER
1192 R2= %2 ;:GENERAL REGISTER
1193 R3= %3 ;:GENERAL REGISTER
1194 R4= %4 ;:GENERAL REGISTER
1195 R5= %5 ;:GENERAL REGISTER
1196 R6= %6 ;:GENERAL REGISTER
1197 R7= %7 ;:GENERAL REGISTER
1198 SP= %6 ;:STACK POINTER
1199 PC= %7 ;:PROGRAM COUNTER
1200
1201 ;*PRIORITY LEVEL DEFINITIONS
1202 PR0= 0 ;:PRIORITY LEVEL 0
1203 PR1= 40 ;:PRIORITY LEVEL 1
1204 PR2= 100 ;:PRIORITY LEVEL 2
1205 PR3= 140 ;:PRIORITY LEVEL 3
1206 PR4= 200 ;:PRIORITY LEVEL 4
1207 PR5= 240 ;:PRIORITY LEVEL 5
1208 PR6= 300 ;:PRIORITY LEVEL 6
1209 PR7= 340 ;:PRIORITY LEVEL 7
1210
1211 ;*"SWITCH REGISTER" SWITCH DEFINITIONS
1212 SW15= 100000
1213 SW14= 40000
1214 SW13= 20000
1215 SW12= 10000
1216 SW11= 4000
1217 SW10= 2000
1218 SW09= 1000
1219 SW08= 400
1220 SW07= 200
1221 SW06= 100
1222 SW05= 40
1223 SW04= 20
1224 SW03= 10
1225 SW02= 4
    
```

1226 000002  
1227 000001  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240 100000  
1241 040000  
1242 020000  
1243 010000  
1244 004000  
1245 002000  
1246 001000  
1247 000400  
1248 000200  
1249 000100  
1250 000040  
1251 000020  
1252 000010  
1253 000004  
1254 000002  
1255 000001  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268 000004  
1269 000010  
1270 000014  
1271 000014  
1272 000014  
1273 000020  
1274 000024  
1275 000030  
1276 000034  
1277 000060  
1278 000064  
1279 000240  
1280  
1281

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  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

.\*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  
.SBTTL MEMORY MANAGEMENT DEFINITIONS

```

1282 ;#KT11 VECTOR ADDRESS
1283
1284 000250 MMVEC= 250
1285
1286 ;#KT11 STATUS REGISTER ADDRESSES
1287
1288 177572 SR0= 177572
1289 177574 SR1= 177574
1290 177576 SR2= 177576
1291 172516 SR3= 172516
1292
1293 ;#USER "I" PAGE DESCRIPTOR REGISTERS
1294
1295 177600 UIPDR0= 177600
1296 177602 UIPDR1= 177602
1297 177604 UIPDR2= 177604
1298 177606 UIPDR3= 177606
1299 177610 UIPDR4= 177610
1300 177612 UIPDR5= 177612
1301 177614 UIPDR6= 177614
1302 177616 UIPDR7= 177616
1303
1304 ;#USER "I" PAGE ADDRESS REGISTERS
1305
1306 177640 UIPAR0= 177640
1307 177642 UIPAR1= 177642
1308 177644 UIPAR2= 177644
1309 177646 UIPAR3= 177646
1310 177650 UIPAR4= 177650
1311 177652 UIPAR5= 177652
1312 177654 UIPAR6= 177654
1313 177656 UIPAR7= 177656
1314
1315 ;#KERNEL "I" PAGE DESCRIPTOR REGISTERS
1316
1317 172300 KIPDR0= 172300
1318 172302 KIPDR1= 172302
1319 172304 KIPDR2= 172304
1320 172306 KIPDR3= 172306
1321 172310 KIPDR4= 172310
1322 172312 KIPDR5= 172312
1323 172314 KIPDR6= 172314
1324 172316 KIPDR7= 172316
1325
1326 ;#KERNEL "I" PAGE ADDRESS REGISTERS
1327
1328 172340 KIPAR0= 172340
1329 172342 KIPAR1= 172342
1330 172344 KIPAR2= 172344
1331 172346 KIPAR3= 172346
1332 172350 KIPAR4= 172350
1333 172352 KIPAR5= 172352
1334 172354 KIPAR6= 172354
1335 172356 KIPAR7= 172356
1336
1337 000700 USESTK =STACK-300

```



1338	000200	CRLF	=200	
1339	076600	MED	=76600	; OPCODE FOR "MED" INSTRUCTION
1340	000114	CACHVEC	=114	; CACHE ERROR INTERRUPT VECTOR
1341	177744	MEMERR	=177744	; CACHE ERROR REGISTER
1342	177746	CCR	=177746	; MEMORY CONTROL REGISTER
1343	177752	HITMIS	=177752	; HIT-MISS REGISTER, A "1"
1344				; IMPLIES A HIT IN CACHE
1345	177766	CPUERR	=177766	; CPU ERROR REGISTER HOLDS CONDITION
1346				; THAT CAUSED TRAP TO ERRVEC (004)
1347	177770	UBREAK	=177770	; ADDRESS OF MICROBREAK REG.
1348				
1349		.EQUIV	SP, KSP	
1350		.EQUIV	SP, USP	
1351		.EQUIV	STACK, KERSTK	
1352		.EQUIV	SR0, MMR0	
1353		.EQUIV	SR2, MMR2	
1354	000000	AC0=	%0	
1355	000001	AC1=	%1	
1356	000002	AC2=	%2	
1357	000003	AC3=	%3	
1358	000004	AC4=	%4	
1359	000005	AC5=	%5	
1360		;LINE CLOCK AND PROGRAMMABLE LINE CLOCK REGISTERS		
1361	172540	PLKCSR=	172540	
1362	172542	PLKCSB=	172542	
1363	000104	PLKVEC=	104	
1364	177546	LKS=	177546	
1365	000100	LKVEC=	100	
1366				
1367		;UNIBUS EXERCISOR REGISTER		
1368	170000	UBEDB=	170000	; DATA BUFFER
1369	170002	UBECC=	170002	; CYCLE COUNT
1370	170004	UBEBA=	170004	; BUS ADDRESS
1371	170006	UBECR1=	170006	; CONTROL REGISTER 1
1372	170010	UBECLR=	170010	; ERROR CLEAR
1373	170014	UBEGO=	170014	; MULTI-EXERCISOR GO
1374	170016	UBECR2=	170016	; CONTROL REGISTER 2
1375	000510	UBEVEC=	510	; INTERRUPT VECTOR
1376				
1377		;MASS BUS TESTER REGISTERS		
1378	160100	MBTCS1=	160100	; CONTROL & STATUS REG. 1
1379	160102	MBTWC=	160102	; WORD COUNT
1380	160104	MBTBA=	160104	; BUS ADDRESS REG.
1381	160106	MBTMR2=	160106	; MAINTENANCE REG. 2
1382	160110	MBTCS2=	160110	; CONTROL & STATUS REG. 2
1383	160112	MBTST=	160112	; STATUS REG. (TAPE)
1384	160114	MBTER=	160114	; ERROR REGISTER
1385	160116	MBTAS=	160116	; ATTENTION SUMMARY REG.
1386	160120	MBTDB=	160120	; DATA BUFFER
1387	160124	MBTMR1=	160124	; MAINTENANCE REG. 1
1388	160126	MBTDT=	160126	; DRIVE TYPE REG.
1389	160174	MBTBAE=	160174	; BUS ADDRESS EXTENSION REG.
1390	160176	MBTCS3=	160176	; CONTROL & STATUS REG. 3
1391	000774	MBTVEC=	774	; INTERRUPT VECTOR
1392	000776	MBTPSW=	776	
1393				

```

1394
1395      100000
1396      040000
1397      020000
1398      002000
1399      001000
1400      000400
1401      000200
1402
1403
1404      000022
1405      000144
1406      000222
1407      000226
1408      000344
1409      000100
1410      000300
1411      000101
1412      000301
1413      000102
1414      000302
1415      000103
1416      000303
1417      000104
1418      000304
1419      000105
1420      000305
1421      000106
1422      000306
1423      000107
1424      000307
1425      000352
1426      000071
1427      000710
1428
1429
1430
1431      000010
1432      000000
1433      140000
1434      000000
1435      030000
1436      000100
1437      000001
1438      000200
1439      000100
1440      000200
1441      000040
1442
1443
1444
1445      000000
1446
1447
1448
1449      000174

```

```

;MISCELLANEOUS BIT ASSIGNMENTS (USED IN OPT.CP)
      KTOPT= 100000      ;BIT15 - BELOW BIT ASSIGNMENTS ARE USED
      EISOPT= 040000      ;BIT14 - IN THE CPCHK ROUTINE
      FPOPT= 020000      ;BIT13 - A BIT FOR EACH OPTION PRESENT
      MBTOPT= 002000      ;BIT10 - BITS 15,14,13,9 NON OPTION BITS
      LKOPT= 001000      ;BIT09 - BITS 12,11,06-00 NOT USED
      TTOPT= 000400      ;BIT08
      UBEOPT= 000200      ;BIT07

;MED OPERATION CODE DEFINITIONS
      RDMHAMI=022
      RDFLAG=144
      WRHAMI=222
      MCNSSW=226
      WRFLAG=344
      RDLJAM=100
      WRLJAM=300
      RDLSERVICE=101
      WRLSERVICE=301
      RDLPBA=102
      WRLPBA=302
      RDLCUA=103
      WRLCUA=303
      RDLFGINT=104
      WRLFGINT=304
      RDLWHAMI=105
      WRLWHAMI=305
      RDLDATA=106
      WRLDATA=306
      RDLTAG=107
      WRLTAG=307
      WRINIT=352
      SMB01=71
      SMB1A=710

```

```

;MICRO ADDR. IN SWAB INST.
;SAME MICRO ADDR. SHIFTED RIGHT

;ADDITIONAL DEFINITIONS
      .EQUIV ERROR,HLT
      CALLHANDLER=10
      KM=0
      UM=140000
      PKM=0
      PUM=30000
      WMP=BIT6
      DPTRP=BIT0
      PABORT=BIT7
      LO=BIT6
      HI=BIT7
      TAG=BITS

.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

```

1450	000174	000000	DISPREG: .WORD 0	::SOFTWARE DISPLAY REGISTER
1451	000176	000000	SWREG: .WORD 0	::SOFTWARE SWITCH REGISTER
1452			.SBTTL STARTING ADDRESS(ES)	
1453	000200	000137 003276	JMP @START	::JUMP TO STARTING ADDRESS OF PROGRAM
1454			.SBTTL ACT11 HOOKS	
1455				
1456			::*****	
1457			;HOOKS REQUIRED BY ACT11	
1458		000204	\$SVPC=.	;SAVE PC
1459		000046	=46	
1460	000046	041230	\$ENDAD	::1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP
1461		000052	=52	
1462	000052	040000	.WORD 40000	::2)SET LOC.52 TO 40000
1463		000204	=\$SVPC	::RESTORE PC



.SBTTL COMMON TAGS

::\*\*\*\*\*  
::\*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
::\*USED IN THE PROGRAM.

1464									
1465									
1466									
1467									
1468									
1469									
1470		001200							
1471	001200		SCMTAG:	.=1200					:: START OF COMMON TAGS
1472	001200	000000	SPASS:	.WORD	0				:: CONTAINS PASS COUNT
1473	001202	000000	STSTM:	.WORD	0				:: CONTAINS THE TEST NUMBER
1474	001204	000	SERFLG:	.BYTE	0				:: CONTAINS ERROR FLAG
1475		001206		.EVEN					
1476	001206	000000	SICNT:	.WORD	0				:: CONTAINS SUBTEST ITERATION COUNT
1477	001210	000000	SLPADR:	.WORD	00				:: CONTAINS SCOPE LOOP ADDRESS
1478	001212	000000	SLPERR:	.WORD	00				:: CONTAINS SCOPE RETURN FOR ERRORS
1479	001214	000000	SERTTL:	.WORD	00				:: CONTAINS TOTAL ERRORS DETECTED
1480	001216	000	SITEMB:	.BYTE	0				:: CONTAINS ITEM CONTROL BYTE
1481	001217	001	SERMAX:	.BYTE	1				:: CONTAINS MAX. ERRORS PER TEST
1482	001220	000000	SERRPC:	.WORD	00				:: CONTAINS PC OF LAST ERROR INSTRUCTION
1483	001222	000000	SGDADR:	.WORD	00				:: CONTAINS ADDRESS OF 'GOOD' DATA
1484	001224	000000	SBDADR:	.WORD	00				:: CONTAINS ADDRESS OF 'BAD' DATA
1485	001226	000000	SGDDAT:	.WORD	00				:: CONTAINS 'GOOD' DATA
1486	001230	000000	SBDAT:	.WORD	00				:: CONTAINS 'BAD' DATA
1487	001232	000000		.WORD	00				:: RESERVED--NOT TO BE USED
1488	001234	000000		.WORD	00				
1489	001236	000	SAUTOB:	.BYTE	0				:: AUTOMATIC MODE INDICATOR
1490	001237	000	SINTAG:	.BYTE	0				:: INTERRUPT MODE INDICATOR
1491	001240	000000		.WORD	0				
1492	001242	177570	SWR:	.WORD	DSWR				:: ADDRESS OF SWITCH REGISTER
1493	001244	177570	DISPLAY:	.WORD	DDISP				:: ADDRESS OF DISPLAY REGISTER
1494	001246	177560	STKS:	177560					:: TTY KBD STATUS
1495	001250	177562	STKB:	177562					:: TTY KBD BUFFER
1496	001252	177564	STPS:	177564					:: TTY PRINTER STATUS REG. ADDRESS
1497	001254	177566	STPB:	177566					:: TTY PRINTER BUFFER REG. ADDRESS
1498	001256	000	SNULL:	.BYTE	0				:: CONTAINS NULL CHARACTER FOR FILLS
1499	001257	002	SFILLS:	.BYTE	2				:: CONTAINS # OF FILLER CHARACTERS REQUIRED
1500	001260	012	SFILLC:	.BYTE	12				:: INSERT FILL CHARS. AFTER A "LINE FEED"
1501	001261	000	STPFLG:	.BYTE	0				:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1502	001262	000000	SREGAD:	.WORD	0				:: CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
1503									
1504	001264	000000	SREG0:	.WORD	0				:: CONTAINS ((\$REGAD)+0)
1505	001266	000000	SREG1:	.WORD	0				:: CONTAINS ((\$REGAD)+2)
1506	001270	000000	SREG2:	.WORD	0				:: CONTAINS ((\$REGAD)+4)
1507	001272	000000	SREG3:	.WORD	0				:: CONTAINS ((\$REGAD)+6)
1508	001274	000000	SREG4:	.WORD	0				:: CONTAINS ((\$REGAD)+10)
1509	001276	000000	SREG5:	.WORD	0				:: CONTAINS ((\$REGAD)+12)
1510	001300	000000	SREG6:	.WORD	0				:: CONTAINS ((\$REGAD)+14)
1511	001302	000000	SREG7:	.WORD	0				:: CONTAINS ((\$REGAD)+16)
1512	001304	000000	STMP0:	.WORD	0				:: USER DEFINED
1513	001306	000000	STMP1:	.WORD	0				:: USER DEFINED
1514	001310	000000	STMP2:	.WORD	0				:: USER DEFINED
1515	001312	000000	STMP3:	.WORD	0				:: USER DEFINED
1516	001314	000000	STMP4:	.WORD	0				:: USER DEFINED
1517	001316	000000	STMP5:	.WORD	0				:: USER DEFINED
1518	001320	000000	STMP6:	.WORD	0				:: USER DEFINED
1519	001322	000000	STMP7:	.WORD	0				:: USER DEFINED

1520	001324	000000		\$TIMES: 0						;; MAX. NUMBER OF ITERATIONS	
1521	001326	000000		\$ESCAPE: 0						;; ESCAPE ON ERROR ADDRESS	
1522	001330	177607	000377	\$BELL: .ASCIZ	<207>	<377>	<377>			;; CODE FOR BELL	
1523	001334	077		\$QUES: .ASCIZ	/?/					;; QUESTION MARK	
1524	001335	015		\$CRLF: .ASCIZ	<15>					;; CARRIAGE RETURN	
1525	001336	000012		\$LF: .ASCIZ	<12>					;; LINE FEED	
1526				:*****							
1527	001340	000000		\$ERRATN: .WORD							
1528	001342	000044		\$FLBUFF: .BLKB	44					;; BUFFER FOR FLOATING POINT CONVERSION	
1529	001406	000000		\$BLUFF: .WORD							
1530	001410	000000		\$ACD: .WORD						;; EXTENDED EXPONENT VALUES	
1531	001412	000000		\$AC1: .WORD						;; FOR THE SIX FLOATING POINT	
1532	001414	000000		\$AC2: .WORD						;; ACCUMULATORS	
1533	001416	000000		\$AC3: .WORD							
1534	001420	000000		\$AC4: .WORD							
1535	001422	000000		\$AC5: .WORD							
1536	001424	000000		\$TMP0: .WORD	0					;; LOCATIONS TO HOLD FLT. PT. OPERANDS	
1537	001426	000000		\$TMP1: .WORD	0						
1538	001430	000000		\$TMP2: .WORD	0						
1539	001432	000000		\$TMP3: .WORD	0						
1540	001434	000000		\$TMP4: .WORD	0						
1541	001436	000000		\$TMP5: .WORD	0						
1542	001440	000000		\$TMP6: .WORD	0						
1543	001442	000000		\$TMP7: .WORD	0						
1544	001444	000000		\$FREG0: .WORD	0						
1545	001446	000000		\$FREG1: .WORD	0						
1546	001450	000000		\$FREG2: .WORD	0						
1547	001452	000000		\$FREG3: .WORD	0						
1548	001454	000000		\$FREG4: .WORD	0						
1549	001456	000000		\$FREG5: .WORD	0						
1550	001460	000000		\$FREG6: .WORD	0						
1551	001462	000000		\$FREG7: .WORD	0						
1552	001464	000004		\$FLTMP0: .BLKW	4					;; FLOATING POINT DBL PREC BUFFER	
1553	001474	000004		\$FLTMP1: .BLKW	4						
1554	001504	001506		\$TKBFRP: .WORD	TKBFR					;; POINTER FOR KEYBOARD BUFFER	
1555	001506	000011		\$TKBFR: .BLKW	11					;; KEYBOARD BUFFER	
1556	001530	000000		\$NOTYPE: .WORD						;; NO TIMEOUT FLAG (INHIBIT WHEN SET)	
1557	001532	000000		\$OPT.CP: .WORD						;; CPU OPTION FLAGS	
1558	001534	000006		\$SRTN: RTT						;; RETURN FOR T-BIT TRAP	
1559	001536	000000		\$VADR: .WORD						;; BUFFER FOR VIRTUAL ADDRESS	
1560	001540	000000		\$PA1500: .WORD						;; BUFFER FOR PHYSICAL ADDRESS BITS<15:00>	
1561	001542	000000		\$PA1716: .WORD						;; PHYSICAL ADDRESS BITS<17:16>	
1562	001544	000		\$NEXEC: .BYTE						;; NO EXECUTE FLAG(NO TEST EXECUTION WHEN SET)	
1563	001545	000		\$MNON: .BYTE						;; MEMORY MGMT FLAG(MGMT IS ON WHEN NON-ZERO)	
1564	001546	000		\$QV: .BYTE						;; QV FLAG(QV PASS WHEN SET)	
1565	001547	000		\$AA: .BYTE						;; AUTO ACCEPT FLAG (AA PASS WHEN SET)	
1566	001550	000000		\$FACTOR: .WORD						;; RELOCATION FACTOR(NUMBER OF	
1567	001552	000000		\$FACTOR: .WORD						;; BYTES ABOVE BASE CODE)	
1568	001554	000000		\$FRSTAD: .WORD						;; FIRST ADDRESS OF SECTION BEING EXECUTED	
1569	001556	000000		\$FRSTMEN: .WORD						;; ADDRESS OF FIRST FREE MEMORY	
1570	001560	000000		\$LSTMEN: .WORD						;; ADDRESS OF LAST FREE MEMORY(IN 28K)	
1571	001562	000000		\$NEXPAR: .WORD						;; NEXT VALUE TO PUT IN PARO	
1572	001564	123456		\$LONUM: .WORD	123456					;; LOW 16 BITS OF RANDOM NUMBER	
1573	001566	065432		\$SHINUM: .WORD	65432					;; HIGH 16 BITS OF RANDOM NUMBER	
1574	001570	001	001	\$NULLS: .BYTE	1,1,1,0					;; BUFFER FOR PRINTER TEST	
1575	001573	000									



```

1576 001574 000060 SUBPASS: .WORD 60 ;SUB-PASS COUNT IN ASCII
1577 001576 000000 SERPSW: .WORD ;ERROR PSW FOR TYPEOUT
1578 001600 000000 EXITFL: .WORD
1579 001602 000000 OLDBASE: .WORD ;SOURCE BASE ADDRESS FOR DEVICE RELOCATION
1580 001604 000000 NMBASL: .WORD ;DEST ADDRESS FOR DEVICE RELOC BITS<15:00>
1581 001606 000000 NMBASH: .WORD ;DEST ADDRESS FOR DEVICE RELOC BITS<21:16>
1582 001610 000000 IOWC: .WORD ;TWO'S COMPLIMENT WORD COUNT FOR DEVICE RELOC
1583 001612 000000 DEVICE: .WORD
1584 001614 000000 DEVINDX: .WORD ;DEVICE INDEX (0 TO 7)
1585 001616 000000 UNITNO: .WORD ;DEVICE UNIT NUMBER
1586 001620 000000 RNTBINX: .WORD ;INDEX TO RUN TABLE
1587 001622 000000 MXMHI: .WORD ;BITS<21:16> OF LAST MEM ADDRESS ON SYSTEM
1588 001624 000000 MXMLO: .WORD ;BITS<15:00> OF LAST MEM ADDRESS ON SYSTEM
1589 001626 000000 RP310: .WORD ;DATA TO LOAD INTO RP03 CS REGISTER
1590 001630 000000 RP311: .WORD ;RP03 FLAG FOR FIRST 2K OF PROGRAM
1591 001632 000000 RK10: .WORD ;DATA TO LOAD INTO RK05 CS REGISTER
1592 001634 000000 RK11: .WORD ;RK05 FLAG FOR FIRST 2K OF PROGRAM
1593 001636 000000 RP411: .WORD ;RP04 FLAG FOR FIRST 2K OF PROGRAM
1594 001640 000000 RS11: .WORD ;RS04 FLAG FOR FIRST 2K OF PROGRAM
1595 001642 000000 MTICKS: .WORD ;ELAPSED RUN TIME IN MINUTES
1596 001644 000000 LTICKS: .WORD ;LOW BYTE=NUMBER OF CLOCK INTERRUPTS (0 TO 59)
1597 ;HIGH BYTE=ELAPSED RUN TIME IN SECONDS(0 TO 59)
1598 001646 000010 SYSSIZE: .BLKW 10 ;SYSTEM SIZE TABLE(ONE ENTRY FOR EACH DEVICE)
1599 001666 000006 RUNTBL: .BLKW 6 ;RUN TIME TABLE(ONE ENTRY FOR EACH 2K BLOCK)
1600 001702 000006 RUNTRAK: .BLKW 6 ;RUN TRACK TABLE(ONE ENTRY FOR EACH 2K BLOCK)
1601 001716 000002 UBESAV: .BLKW 2 ;BASE ADDRESS OF UBE TRANSFER IN PROGRESS
1602 001722 000002 UBEADR: .BLKW 2 ;ADDRESS THAT GETS LOADED INTO UBE BA REG
1603 001726 000002 ERRBA: .BLKW 2 ;18 BIT UNIBUS ADDRESS WHEN DEVICE DETECTED AN ERROR
1604 001732 000000 MEMFLG: .WORD 0 ;HOLDS FLAG FOR UBE & MBT
1605 001734 000000 ERRADR: .WORD 0 ;HOLDS ADDR. FOR UBE & MBT
1606 001736 000000 REPPSW: .WORD 0 ;HOLDS PSW FOR ERROR REPORT
1607 001740 000000 REPREG: .WORD 0 ;HOLDS REGISTER FOR ERROR REPORT
1608 001742 000000 REPADR: .WORD 0 ;HOLDS LOOP ADDR. FOR ERROR REPROT
1609 001744 000000 REPMR0: .WORD 0 ;HOLDS MMR0 FOR ERROR REPORT
1610 001746 000000 REPMR2: .WORD 0 ;HOLDS MMR2 FOR ERROR REPORT
1611 001750 000000 SRCADR: .WORD 0 ;HOLDS ADDR. DURING DATA CHECK
1612 .SBTTL DEVICE HANDLER STATUS WORDS
1613 ;* EACH WORD HAS THE FOLLOWING BIT ASSIGNMENTS:
1614 ;* 7 HANDLER READY
1615 ;* 8 REPEAT LAST FUNCTION
1616 ;* 15 ERROR
1617 001752 000200 RP3HSTAT: .WORD 200 ;RP03
1618 001754 000200 RKHSTAT: .WORD 200 ;RK05
1619 001756 000200 SPARE0: .WORD 200
1620 001760 000200 SPARE1: .WORD 200
1621 001762 000200 RP4HSTAT: .WORD 200 ;RP04
1622 001764 000200 RSHSTAT: .WORD 200 ;RS04
1623 001766 000200 .WORD 200 ;SPARE
1624 001770 000200 .WORD 200 ;SPARE
1625
1626 .SBTTL DEVICE HANDLER WORD COUNTS
1627 ;* THIS TABLE GETS LOADED BY THE I/O
1628 ;* RELOCATION ROUTINE WITH THE TWO'S COMPLIMENT WORD
1629 ;* COUNT FOR THE TRANSFER FOR THE PARTICULAR DEVICE.
1630 001772 000000 RP3HWC: .WORD ;RP03
1631 001774 000000 RKHWC: .WORD ;RK05

```



1632	001776	000000		.WORD	; SPARE
1633	002000	000000		.WORD	; SPARE
1634	002002	000000	RP4HWC:	.WORD	; RP04
1635	002004	000000	RSHWC:	.WORD	; RS04
1636					
1637			.SBTTL	DEVICE HANDLER OLD BASE ADDRESS	
1638			.*	THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE	
1639			.*	WITH THE BASE ADDRESS OF THE SOURCE DATA FOR THE	
1640			.*	DEVICE THAT IS GOING TO TRANSFER THE DATA.	
1641	002006	000000	RP3OLD:	.WORD	; RP03
1642	002010	000000		.WORD	
1643	002012	000000	RKOLD:	.WORD	; RK05
1644	002014	000000		.WORD	
1645	002016	000000		.WORD	; SPARE
1646	002020	000000		.WORD	; SPARE
1647	002022	000000		.WORD	
1648	002024	000000		.WORD	; SPARE
1649	002026	000000	RP4OLD:	.WORD	; RP04
1650	002030	000000		.WORD	
1651	002032	000000	RSOLD:	.WORD	; RS04
1652	002034	000000		.WORD	
1653					
1654			.SBTTL	DEVICE HANDLER NEW BASE ADDRESSES	
1655			.*	THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE	
1656			.*	WITH THE BASE ADDRESS OF THE DESTINATION FOR THE	
1657			.*	PARTICULAR DEVICE THAT IS GOING TO DO THE TRANSFER.	
1658	002036	000000	RP3NWL:	.WORD	; RP03
1659	002040	000000	RP3NWH:	.WORD	
1660	002042	000000	RKNEWL:	.WORD	; RK05
1661	002044	000000	RKNEWH:	.WORD	
1662	002046	000000		.WORD	; SPARE
1663	002050	000000		.WORD	
1664	002052	000000		.WORD	; SPARE
1665	002054	000000		.WORD	
1666	002056	000000	RP4NWL:	.WORD	; RP04
1667	002060	000000	RP4NWH:	.WORD	
1668	002062	000000	RSNEWL:	.WORD	; RS04
1669	002064	000000	RSNEWH:	.WORD	
1670					
1671			.SBTTL	DEVICE HANDLER UNIT NUMBER	
1672			.*	THIS TABLE GETS LOADED BY THE I/O RELOCATION ROUTINE.	
1673			.*	IT TELLS THE DEVICE HANDLER WHICH UNIT NUMBER IS	
1674			.*	TO DO THE TRANSFER.	
1675	002066	000000	RP3UNIT:	.WORD	; RP03
1676	002070	000000	RKUNIT:	.WORD	; RK05
1677	002072	000000		.WORD	; SPARE
1678	002074	000000		.WORD	; SPARE
1679	002076	000000	RP4UNIT:	.WORD	; RP04
1680	002100	000000	RSUNIT:	.WORD	; RS04
1681					
1682			.SBTTL	ADDRESS OF THE DEVICE HANDLERS	
1683			.*	THIS TABLE CONTAINS THE ADDRESS OF THE DEVICE HANDLER	
1684			.*	ROUTINES. IT IS USED BY THE I/O RELOCATION ROUTINE	
1685			.*	TO TRANSFER CONTROL TO THE DEVICE HANDLER.	
1686	002102	041250	RP3HANA:	.WORD	RP3DRV ; RP03
1687	002104	041644	RKHANA:	.WORD	RKDRV ; RK05

1688	002106	000000		.WORD	: SPARE
1689	002110	000000		.WORD	: SPARE
1690	002112	042220	RP4HANA:	.WORD	: RP04
1691	002114	042614	RSHANA:	.WORD	: RS04
1692					
1693			.SBTTL	DEVICE HANDLER DISK ADDRESS TABLE	
1694			:::	THIS TABLE GETS LOADED BY THE DEVICE HANDLER WITH THE	
1695			:::	DISK ADDRESS(SECTOR AND CYLINDER) OF THE CURRENT	
1696			:::	TRANSFER.	
1697	002116	000000	RP3HDA:	.WORD	: RP03 DISK ADDRESS
1698	002120	000000	RP3HDC:	.WORD	: RP03 DESIRED CYLINDER
1699	002122	000000	RKHDA:	.WORD	: RK05 DISK ADDRESS
1700	002124	000000		.WORD	: SPARE
1701	002126	000000	RP4HDA:	.WORD	
1702	002130	000000	RP4HDC:	.WORD	: RP04 DESIRED CYLINDER
1703	002132	000000	RSHDA:	.WORD	: RS04 DISK ADDRESS
1704					
1705			.SBTTL	DEVICE HANDLER FUNCTION TABLE	
1706			:::	THIS TABLE GETS LOADED BY THE DEVICE HANDLERS	
1707			:::	AND THE DEVICE SERVICE ROUTINES. IT TELLS THE ROUTINES	
1708			:::	WHICH FUNCTION TO DO NEXT.	
1709	002134	000000	RP3FUN:	.WORD	: RP03
1710	002136	000000	RKFUN:	.WORD	: RK05
1711	002140	000000		.WORD	: SPARE
1712	002142	000000	RP4FUN:	.WORD	: RP04
1713	002144	000000	RSFUN:	.WORD	: RS04
1714					
1715			.SBTTL	DEVICE HANDLER RETRY COUNT	
1716			:::	THIS TABLE GETS LOADED BY THE DEVICE HANDLERS AND IS USED	
1717			:::	BY THE DEVICE SERVICE ROUTINES. IF AN ERROR OCCURS	
1718			:::	THE DEVICE SERVICE ROUTINE WILL RETRY THE FUNCTION UNTIL	
1719			:::	THE BYTE IN THIS TABLE GOES TO ZERO. IT IS INITIALIZED	
1720			:::	TO A -3.	
1721	002146	000	RP3TRY:	.BYTE	: RP03
1722	002147	000	RKTRY:	.BYTE	: RK05
1723	002150	000		.BYTE	: SPARE
1724	002151	000	RP4TRY:	.BYTE	: RP04
1725	002152	000	RSTRY:	.BYTE	: RS04
1726		002154		.EVEN	
1727					
1728			.SBTTL	DEVICE REGISTER TABLES	
1729			:::	THE FOLLOWING TABLES CONTAIN THE STANDARD ADDRESS FOR	
1730			:::	THE DEVICES USED BY THIS PROGRAM. IF A DEVICE IS PLACED	
1731			:::	AT A NON-STANDARD ADDRESS THE APPROPRIATE TABLE CAN BE	
1732			:::	CHANGED AND THE PROGRAM WILL OPERATE THAT DEVICE.	
1733			:::		
1734			:::	EXCEPTION--SEE DOCUMENTATION FOR RP03 AND RP04 PROBLEMS.	
1735			.SBTTL	RP11/RP03 REGISTERS	
1736	002154	176710	RP3DS:	.WORD	176710 : DRIVE STATUS
1737	002156	176712	RP3ER:	.WORD	176712 : ERROR REGISTER
1738	002160	176714	RP3CS:	.WORD	176714 : CONTROL AND STATUS
1739	002162	176716	RP3WC:	.WORD	176716 : WORD COUNT
1740	002164	176720	RP3BA:	.WORD	176720 : BUS ADDRESS
1741	002166	176724	RP3DA:	.WORD	176724 : DISK ADDRESS
1742	002170	176722	RP3DC:	.WORD	176722 : DESIRED CYLINDER
1743	002172	000254	RP3VEC:	.WORD	254 : INTERRUPT VECTOR



1744 002174 000256  
1745  
1746  
1747 002176 177400  
1748 002200 177402  
1749 002202 177404  
1750 002204 177406  
1751 002206 177410  
1752 002210 177412  
1753 002212 000220  
1754 002214 000222  
1755  
1756  
1757 002216 176700  
1758 002220 176702  
1759 002222 176704  
1760 002224 176700  
1761 002226 176706  
1762 002230 176710  
1763 002232 176752  
1764 002234 176712  
1765 002236 176714  
1766 002240 176734  
1767 002242 176740  
1768 002244 176742  
1769 002246 176736  
1770 002250 176732  
1771 002252 000254  
1772 002254 000256  
1773  
1774  
1775 002256 172040  
1776 002260 172042  
1777 002262 172044  
1778 002264 172040  
1779 002266 172046  
1780 002270 172050  
1781 002272 172072  
1782 002274 172052  
1783 002276 172054  
1784 002300 000204  
1785 002302 000206  
1786  
1787  
1788  
1789  
1790  
1791  
1792 002304 170002  
1793 002306 170004  
1794 002310 170016  
1795 002312 170006  
1796 002314 170010  
1797 002316 000510  
1798 002320 000512  
1799

RP3PSW: .WORD 256 ; INTERRUPT VECTOR+2

.SBTTL RK11/RK05 REGISTERS

RKDS: .WORD 177400 ; DRIVE STATUS  
RKER: .WORD 177402 ; ERROR REGISTER  
RKCS: .WORD 177404 ; CONTROL AND STATUS  
RKWC: .WORD 177406 ; WORD COUNT  
RKBA: .WORD 177410 ; BUS ADDRESS  
RKDA: .WORD 177412 ; DISK ADDRESS  
RKVEC: .WORD 220 ; INTERRUPT VECTOR  
RKPSW: .WORD 222 ; INTERRUPT VECTOR+2

.SBTTL RH11/RP04 REGISTERS

RP4CS1: .WORD 176700 ; CONTROL AND STATUS #1  
RP4WC: .WORD 176702 ; WORD COUNT  
RP4BA: .WORD 176704 ; BUS ADDRESS  
RP4BAE: .WORD 176700 ; CONTROL AND STATUS #1  
RP4DA: .WORD 176706 ; DISK ADDRESS  
RP4CS2: .WORD 176710 ; CONTROL AND STATUS #2  
RP4CS3: .WORD 176752 ; CONTROL AND STATUS #3  
RP4DS: .WORD 176712 ; DRIVE STATUS  
RP4ER1: .WORD 176714 ; ERROR REG #1  
RP4DC: .WORD 176734 ; DESIRED CYLINDER  
RP4ER2: .WORD 176740 ; ERROR REG #2  
RP4ER3: .WORD 176742 ; ERROR REG #3  
RPCC: .WORD 176736 ; CURRENT CYLINDER  
RP4OF: .WORD 176732 ; OFFSET REGISTER  
RP4VEC: .WORD 254 ; INTERRUPT VECTOR  
RP4PSW: .WORD 256 ; INTERRUPT VECTOR+2

.SBTTL RH11/RS04 REGISTERS

RSCS1: .WORD 172040 ; CONTROL AND STATUS #1  
RSWC: .WORD 172042 ; WORD COUNT  
RSBA: .WORD 172044 ; BUS ADDRESS  
RSBAE: .WORD 172040 ; CONTROL AND STATUS #1  
RSDA: .WORD 172046 ; DISK ADDRESS  
RSCS2: .WORD 172050 ; CONTROL AND STATUS #2  
RSCS3: .WORD 172072 ; CONTROL AND STATUS #3  
RSDS: .WORD 172052 ; DRIVE STATUS  
RSER: .WORD 172054 ; ERROR REG  
RSVEC: .WORD 204 ; INTERRUPT VECTOR  
RSPSW: .WORD 206 ; INTERRUPT VECTOR+2

.SBTTL UNIBUS EXERCISER REGISTER ADDRESS TABLE  
:\* THIS TABLE IS ASSEMBLED FOR UBE #0. IF THE UBE  
:\* ADDRESSES ARE CUT FOR OTHER THAN UNIT #0, THE PROGRAM  
:\* WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A  
:\* UBE AT ADDRESSES 770000, 770020, 770040, AND 770060.

UBETBL: .WORD UBECC ; CYCLE COUNT  
.WORD UBEBA ; BUS ADDRESS REG  
.WORD UBECA2 ; CONTROL REGISTER #2  
.WORD UBECA1 ; CONTROL REGISTER #1  
.WORD UBECLA ; UBE CLEAR ADDRESS  
.WORD UBEVEC ; INTERRUPT VECTOR  
.WORD UBEVEC+2 ; INTERRUPT VECTOR +2



1800  
1801  
1802  
1803  
1804  
1805 002322 160100  
1806 002324 160102  
1807 002326 160104  
1808 002330 160100  
1809 002332 160106  
1810 002334 160110  
1811 002336 160112  
1812 002340 160114  
1813 002342 160176  
1814 002344 000774  
1815 002346 000776  
1816 002350 160126  
1817 002352 160200  
1818 002354 160300  
1819 002356 160400  
1820  
1821  
1822 002360 000000  
1823 002362 000000  
1824 002364 000000  
1825 002366 000000  
1826 002370 000040  
1827 002470 000000  
1828 002472 000000  
1829 002474 000000  
1830 002476 000000  
1831 002500 000000  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847 002502  
1848  
1849  
1850 002502  
1851 002502 201 001  
1852 002504 202 002  
1853 002506 203 003  
1854 002510 204 004  
1855 002512 205 005

```
.SBTTL MASS BUS TESTER REGISTER ADDRESSES
:* THE PROGRAM IS ASSEMBLED WITH ADDRESSES FOR A MBT
:* AT 760100. IF THE MBT IS AT ANOTHER ADDRESS THE PROGRAM
:* WILL CHANGE THIS TABLE. THE PROGRAM LOOKS FOR A MBT
:* AT ADDRESSES 760100, 760200, 760300, AND 760400.
MBTTBL: .WORD MBTCS1 :CONTROL AND STATUS #1
        .WORD MBTWC :WORD COUNT
        .WORD MBTBA :BUS ADDRESS
        .WORD MBTCS1 :CONTROL AND STATUS #1
        .WORD MBTMR2 :MAINTENANCE REGISTER #2
        .WORD MBTCS2 :CONTROL REGISTER #2
        .WORD MBTST :STATUS REGISTER
        .WORD MBTER :ERROR REGISTER
        .WORD MBTCS3 :CONTROL REGISTER #3
        .WORD MBTVEC :INTERRUPT VECTOR
        .WORD MBTPSW :INTERRUPT VECTOR+2
        .WORD MBTDT :DRIVE TYPE REGISTER
MBTN2: .WORD 160200 :MASS BUS TESTER #2
MBTN3: .WORD 160300 :MASS BUS TESTER #3
MBTN4: .WORD 160400 :MASS BUS TESTER #4
```

```
.SBTTL MED TEST TABLES
TLOC1: .WORD 0
PSMHOL: .WORD 0
TABBEG: .WORD 0
TABEND: .WORD 0
STGBLK: .BLKW 40
TLOC2: .WORD 0
MEDTP0: .WORD 0
MEDTP1: .WORD 0
MEDTP2: .WORD 0
MEDTP3: .WORD 0
```

```
:*
:* TABLE II
:*
:* FOLLOWING IS A TABLE OF INTERNAL REGISTER OPERATION CODES
:* USED FOR TESTING THE MED INSTRUCTION. LABELS CORRESPOND
:* TO REGISTER NAMES. THE HIGH BYTE IS THE READ OPERATION
:* CODE, THE LOW BYTE THE WRITE CODE.
:* NOTE: WHEN ADDING OR DELETING
:* ENTRIES IN THIS TABLE, CHECK DUAL
:* ADDRESSING TEST TO SEE THAT THE "SCRATCH
:* PAD LIMITS" ARE MAINTAINED.
```

```
TBL2:
ASP1:
R1A: .BYTE 201,001 ;A SCRATCH PAD - LO
R2A: .BYTE 202,002 ;LOBYTE, HIBYTE=WRITE CODE, READ CODE
R3A: .BYTE 203,003
R4A: .BYTE 204,004
RSA: .BYTE 205,005
```

MED TEST TABLES

1856	002514	206	006
1857	002516	210	010
1858	002520	211	011
1859	002522	212	012
1860	002524	213	013
1861	002526	214	014
1862	002530	215	015
1863	002532	216	016
1864	002534	217	017
1865	002536	220	020
1866	002540	221	021
1867	002542	222	022
1868	002544	223	023
1869	002546	224	024
1870	002550	225	025
1871	002552	226	026
1872	002554	227	027
1873	002556	230	030
1874	002560	231	031
1875	002562	232	032
1876	002564	233	033
1877	002566	234	034
1878	002570	235	035
1879	002572	236	036
1880	002574	237	037
1881			
1882			
1883	002576		
1884	002576	241	041
1885	002600	242	042
1886	002602	243	043
1887	002604	244	044
1888	002606	245	045
1889	002610	246	046
1890	002612	250	050
1891	002614	251	051
1892	002616	252	052
1893	002620	253	053
1894	002622	254	054
1895	002624	255	055
1896	002626	256	056
1897	002630	257	057
1898	002632	260	060
1899	002634	261	061
1900	002636	262	062
1901	002640	263	063
1902	002642	264	064
1903	002644	265	065
1904	002646	266	066
1905	002650	270	070
1906	002652	272	072
1907	002654	273	073
1908	002656	274	074
1909	002660	275	075
1910	002662	276	076
1911	002664	277	077

R6A:	.BYTE	206,006
FAC3.0:	.BYTE	210,010
FAC3.1:	.BYTE	211,011
FAC3.2:	.BYTE	212,012
FAC3.3:	.BYTE	213,013
FAC3.4:	.BYTE	214,014
FAC3.5:	.BYTE	215,015
UR6A:	.BYTE	216,016
FDST3:	.BYTE	217,017
WCSA.0:	.BYTE	220,020
WCSA.1:	.BYTE	221,021
GNHAM:	.BYTE	222,022
CNSTSM:	.BYTE	223,023
RT1A:	.BYTE	224,024
RT2A:	.BYTE	225,025
CNSSM:	.BYTE	226,026
CNSCOR:	.BYTE	227,027
FAC1.0:	.BYTE	230,030
FAC1.1:	.BYTE	231,031
FAC1.2:	.BYTE	232,032
FAC1.3:	.BYTE	233,033
FAC1.4:	.BYTE	234,034
FAC1.5:	.BYTE	235,035
FPSHI:	.BYTE	236,036
ASP2:	FDST1:	.BYTE 237,037

;A SCRATCH PAD-HI

BSP1:		
R18:	.BYTE	241,041
R28:	.BYTE	242,042
R38:	.BYTE	243,043
R48:	.BYTE	244,044
R58:	.BYTE	245,045
R68:	.BYTE	246,046
FAC2.0:	.BYTE	250,050
FAC2.1:	.BYTE	251,051
FAC2.2:	.BYTE	252,052
FAC2.3:	.BYTE	253,053
FAC2.4:	.BYTE	254,054
FAC2.5:	.BYTE	255,055
UR6B:	.BYTE	256,056
FDST2:	.BYTE	257,057
WCSB.0:	.BYTE	260,060
WCSB.1:	.BYTE	261,061
WCSADR:	.BYTE	262,062
RZERO:	.BYTE	263,063
RT1B:	.BYTE	264,064
RT2B:	.BYTE	265,065
RVECT:	.BYTE	266,066
FACO.0:	.BYTE	270,070
FACO.1:	.BYTE	272,072
FACO.2:	.BYTE	273,073
FACO.4:	.BYTE	274,074
FACO.5:	.BYTE	275,075
FEA:	.BYTE	276,076
BSP2:	FDST0:	.BYTE 277,077

;B SCRATCH PAD - LO

;B SCRATCH PAD - HI



```

1912
1913
1914 002666
1915 002666 300 100
1916 002670 301 101
1917 002672 302 102
1918 002674 303 103
1919 002676 304 104
1920 002700 305 105
1921 002702 307 107
1922 002704 310 110
1923 002706 311 111
1924 002710 312 112
1925 002712 313 113
1926 002714 316 116
1927
1928 002716 000000
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939 002720
1940 002720 200 000
1941 002722 207 007
1942 002724 240 040
1943 002726 247 047
1944 002730 314 114
1945 002732 317 117
1946
1947
1948 002734
1949
1950 002734 306
1951 002735 106
1952 002736 315
1953 002737 115
1954 002740 267
1955 002741 067
1956 002742 140
1957 002743 141
1958 002744 142
1959 002745 143
1960 002746 344
1961 002747 144
1962 002750 345
1963 002751 146
1964 002752 346
1965 002753 147
1966 002754 347
1967 002755 351
    
```

```

CSP1:
LJAM: .BYTE 300,100 ;C SCRATCH PAD
LSERV: .BYTE 301,101
LPBA: .BYTE 302,102
LCUA: .BYTE 303,103
LFGIN: .BYTE 304,104
LHAM: .BYTE 305,105
LTAG: .BYTE 307,107
CNSCO: .BYTE 310,110
CNSC1: .BYTE 311,111
CNSC2: .BYTE 312,112
CST200: .BYTE 313,113
CSP2: CNSTO: .BYTE 316,116

.WORD 0 ;END OF ADDRESS TABLE = 0
    
```

```

;*
;* TABLE IV
;*
;* THE LIST BELOW CONTAINS THOSE OPERATION CODES
;* CORRESPONDING TO THE INTERNAL REGISTERS WHICH MUST
;* BE TESTED SEPERATELY BECAUSE THEY ARE READ-ONLY,
;* WRITE-ONLY, OR USED IN MACRO CODE EXECUTION, ETC. . .
;*
    
```

```

TBL4:
ROA: .BYTE 200,000 ;LOBYTE, HYBYTE - WRITE CODE, READ CODE
R7A: .BYTE 207,007
ROB: .BYTE 240,040
R7B: .BYTE 247,047
CNST2: .BYTE 314,114
CNST1: .BYTE 317,117
;*
;* TABLE V
    
```

```

TBL5:
LCDTA: .BYTE 306 ;THIS TABLE CONTAINS THE OPERATION
;CODES OF THOSE INTERNAL REGISTERS
;WHICH MUST BE TESTED USING THE
;MICROBREAK REGISTER. THEIR
;ASSOCIATED MICRO-ADDRESSES ARE IN
;THE NEXT TABLE
MD: .BYTE 315
CNSCTL: .BYTE 267
;THESE MICROADDRESSES ARE THE ENTRY
;POINTS INTO THE MICROCODE FOR THE SERVICE
;OF THE INTERNAL REGISTERS
JAM: .BYTE 140
SERV: .BYTE 141
PBA: .BYTE 142
CUA: .BYTE 143
FLAG: .BYTE 344
DREG: .BYTE 345
REV: .BYTE 146
SREG: .BYTE 346
COUNT: .BYTE 147
NUA: .BYTE 347
RES: .BYTE 351
    
```



1968 002756 152  
 1969 002757 352  
 1970 002760 153  
 1971 002761 000  
 1972  
 1973  
 1974  
 1975  
 1976 002762  
 1977  
 1978 002762 003330  
 1979 002764 003150  
 1980 002766 003375  
 1981 002770 003271  
 1982 002772 003240  
 1983 002774 003224  
 1984 002776 003160  
 1985 003000 003161  
 1986 003002 003170  
 1987 003004 003171  
 1988 003006 003344  
 1989 003010 003320  
 1990 003012 003345  
 1991 003014 003340  
 1992 003016 003350  
 1993 003020 003341  
 1994 003022 003351  
 1995 003024 003355  
 1996 003026 003720  
 1997 003030 003724  
 1998 003032 003721  
 1999  
 2000  
 2001  
 2002  
 2003  
 2004 003034  
 2005  
 2006 003034 000100 077600  
 2007 003040 000101 000010  
 2008 003044 000102 020000  
 2009 003050 000103 000004  
 2010 003054 000104 050000  
 2011 003060 000105 054000  
 2012 003064 000107 024000  
 2013 003070 000110 177400  
 2014 003074 000111 177600  
 2015 003100 000112 100000  
 2016 003104 000113 000200  
 2017 003110 000114 000002  
 2018 003114 000116 000000  
 2019 003120 000117 000001  
 2020 003124 000000

DCSO: .BYTE 152  
 INIT: .BYTE 352  
 DCS1: .BYTE 153  
 .BYTE 0

;0 BYTE TERMINATES THE PROGRAM

;\* .EVEN  
 ;\* TABLE VI  
 ;\*

TBL6:  
 ULCDTA: .WORD 3330  
 .WORD 3150  
 UMD: .WORD 3375  
 .WORD 3271  
 UCNSCTL: .WORD 3240  
 .WORD 3224  
 UJAM: .WORD 3160  
 USERV: .WORD 3161  
 UPBA: .WORD 3170  
 UCUA: .WORD 3171  
 UFLAG: .WORD 3344  
 .WORD 3320  
 UDREG: .WORD 3345  
 UREV: .WORD 3340  
 USREG: .WORD 3350  
 UCOUNT: .WORD 3341  
 UNUA: .WORD 3351  
 URES: .WORD 3355  
 UDCSO: .WORD 3720  
 UINIT: .WORD 3724  
 UDCS1: .WORD 3721

; THIS TABLE CONTAINS THE MICRO-ADDRESSES  
 ; WHICH ARE LOADED INTO THE MICROBREAK  
 ; REG. TO TEST THE OPERATION CODES  
 ; CONTAINED IN THE PRECEEDING TABLE.

; THESE MICROADDRESSES SHOULD BE  
 ; CHANGED (IF NECESSARY) IF THE BASE MACHINE  
 ; MICROCODE SHIFTS (MICROCODE SERVICING  
 ; THE INTERNAL REGISTERS).

;\* TABLE VII  
 ;\*

;\* THIS TABLE HOLDS THE OPERATION CODES AND THE CONSTANT  
 ;\* VALUE EXPECTED FOR CERTAIN INTERNAL REGISTERS.  
 ;\*

TBL7:  
 CLJAM: .WORD 100,77600  
 CLSERV: .WORD 101,10  
 CLPBA: .WORD 102,20000  
 CLCUA: .WORD 103,4  
 CLFGIN: .WORD 104,50000  
 CLWHAM: .WORD 105,54000  
 CLTAG: .WORD 107,24000  
 CCNSCO: .WORD 110,177400  
 CCNSC1: .WORD 111,177600  
 CCNSC2: .WORD 112,100000  
 CCST200: .WORD 113,200  
 CCNST2: .WORD 114,2  
 CCNST0: .WORD 116,0  
 CCNST1: .WORD 117,1  
 .WORD 0

.SBTTL ERROR POINTER TABLE

;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
;\*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;\*NOTE1: IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).  
;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;\* EM ;;POINTS TO THE ERROR MESSAGE  
;\* DH ;;POINTS TO THE DATA HEADER  
;\* DT ;;POINTS TO THE DATA  
;\* DF ;;POINTS TO THE DATA FORMAT

SERRTB:

2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076

003126  
003126 056411  
003130 056436  
003132 056502  
003134 056475  
003136 056514  
003140 056542  
003142 056570  
003144 056475  
003146 056600  
003150 056635  
003152 056704  
003154 056475  
003156 056720  
003160 056747  
003162 057014  
003164 057026  
003166 057032  
003170 057071  
003172 057130  
003174 057125  
003176 057140  
003200 057210  
003202 057230  
003204 057026  
003206 000000  
003210 000000  
003212 000000  
003214 000000  
003216 057236  
003220 057307  
003222 057362  
003224 057356

:ITEM 1  
EM1 ;UNEXPECTED TRAP TO 4  
DH1 ;VIRTPC PHYSPC PSW CPUERR  
DT1 ;VADR,VADR,REPPSW,REPREG  
DF1 ;0,1,0,0,0  
:ITEM 2  
EM2 ;UNEXPECTED TRAP TO 10  
DH2 ;VIRTPC PHYSPC PSW  
DT2 ;VADR,VADR,REPPSW  
DF1  
:ITEM 3  
EM3 ;UNEXPECTED TRAP TO 250(MGMT)  
DH3 ;VIRTPC PHYSPC PSW MMRO MMR2  
DT3 ;VADR,VADR,REPPSW,REPMRO,REPMR2  
DF1  
:ITEM 4  
EM4 ;UNEXPECTED TRAP TO 114  
DH4 ;VIRTPC PHYSPC PSW MEMERRREG  
DT4 ;VADR,VADR,REPPSW,REPREG  
DF4 ;0,1,0,0  
:ITEM 5  
EM5 ;PARITY ERROR DURING DATA CHECK  
DH5 ;SRCADR DSTADR MEM ERR REG  
DT5 ;SRCADR,PA1500,REPREG  
DF5  
:ITEM 6  
EM6 ;ERROR DURING CHECK OF RELOCATED DATA  
DH6 ;SRCADR DSTADR  
DT6 ;STMP0,PA1500  
DF4  
:ITEM 7  
0 ;DEVICE ERROR  
0  
0  
0  
:ITEM 10  
EM10 ;ERROR DURING DATA CHECK-RELOC WAS BY I/O  
DH10 ;SRCADR DSTADR DEVICE THAT DID XFER  
DT10 ;STMP0,VADR,STMP2,STMP3  
DF10 ;0,1,3,0  
:ITEM 11

# E05

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 56

## ERROR POINTER TABLE

2077	003226	057637	EM14	; FLOATING POINT ERROR	
2078	003230	057374	DH11	;	DATA1 DATA2
2079	003232	057472	DT11	; FLTMP0, FREG2, FLTMP1, FREG3	
2080	003234	057465	DF11	; 5,0,5,0	
2081			; ITEM 12		
2082	003236	057504	EM12	; UNIBUS EXERCISOR NON-EXISTANT MEMORY	
2083	003240	057542	DH12	; PHYSICAL ADDRESS	
2084	003242	057560	DT12	; PA1500	
2085	003244	057556	DF12	; 2	
2086			; ITEM 13		
2087	003246	057564	EM13	; MASS BUS TESTER NON-EXISTANT MEMORY	
2088	003250	057622	DH13	; PHYSICAL ADDRESS	
2089	003252	057560	DT12		
2090	003254	057556	DF12		
2091			; ITEM 14		
2092	003256	057637	EM14	; FLOATING POINT ERROR	
2093	003260	057664	DH14	;	DATA1 DATA2
2094	003262	057704	DT14	; FTMP4, FREG2, FTMP6, FREG3	
2095	003264	057716	DF14	; 4,0,4,0	
2096			; ITEM 15		
2097	003266	057722	EM15	; DEVICE HUNG	
2098	003270	000000	0		
2099	003272	000000	0		
2100	003274	000000	0		
2101					



F05

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 57  
DQKDC.A.P11 07-FEB-77 09:58 PROGRAM INITIALIZATION

2102  
2103  
2104

.SBTTL PROGRAM INITIALIZATION

;\*\*\*\*\*

2  
A

```

2105 003276 012706 001200 START: MOV #KERSTK,SP ;SET KERNEL STACK PTR
2106 003302 012737 017654 001566 MOV #17654,#SHINUM ;INITIALIZE RANDOM NUM GEN
2107 003310 012737 123456 001564 MOV #123456,#SLONUM
2108
2109 ;DETERMINE HOW PROGRAM WAS LOADED AND WHAT MODE (IF ACT11)
2110 ;AND SET MEMORY PROTECTION.
2111 003316 005037 001546 CLR #QV ;SET NOT QV NOR AA MODE
2112 003322 005027 CLR (PC)+ ;SET NOT XXDP
2113 003324 000 .BYTE 0 ;XXDP INDICATOR
2114 003325 000 .BYTE 0 ;XXDP CHAIN MODE INDICATOR
2115 003326 005027 CLR (PC)+ ;CLEAR MEMORY PROTECTION LIMIT
2116 003330 000000 PROT: .WORD 0 ;WILL CONTAIN MEM PROT LIMIT
2117 003332 005737 041234 TST #SENDAD+4 ;BRANCH IF NOT QV
2118 003336 100003 BPL 1$
2119 003340 110637 001546 MOVB SP,#QV ;SET ACT11 QV MODE
2120 003344 000414 BR 3$
2121
2122 003346 001003 1$: BNE 2$
2123 003350 110637 001547 MOVB SP,#AA ;SET ACT11 AA MODE
2124 003354 000410 BR 3$
2125
2126 003356 113737 000041 003324 2$: MOVB #41,#XXDP ;GET LOAD MEDIA
2127 003364 005737 000042 TST #42 ;BRANCH IF NOT IN CHAIN MODE
2128 003370 001402 BEQ 3$
2129 003372 110637 003325 MOVB SP,#XXDPC ;SET CHAIN MODE INDICATOR
2130
2131 ;SET MEMORY PROTECTION LIMITS
2132 003376 005737 001546 3$: TST #QV ;BRANCH IF QV OR AA
2133 003402 001006 BNE MEMSIZ
2134 003404 005737 003324 TST #XXDP ;BRANCH IF NOT VIA XXDP
2135 003410 001403 BEQ MEMSIZ
2136 003412 012737 005700 003330 MOV #5700,#PROT ;PROTECT XXDP MONITOR
2137 003420 005000 MEMSIZ: CLR R0 ;INITIALIZE R0 WITH FIRST ADDR.
2138 003422 012737 003434 000004 MOV #11$,#ERRVEC ;SET TIMEOUT VECTOR TO 11$
2139 003430 005720 10$: TST (R0)+ ;DOES ADDR. IN R0 EXIST?
2140 003432 000776 BR 10$ ;CONTINUE UNTIL AN ADDR. TIMES OUT
2141
2142 003434 022626 11$: CMP (SP)+,(SP)+ ;CLEAN UP STACK
2143 003436 012737 055064 000004 MOV #ERRPT,#ERRVEC ;RESTORE TIMEOUT SERVICE ROUTINE
2144 003444 162700 000004 SUB #4,R0 ;GET ADDR. OF LAST MEM. LOC.
2145 003450 010037 001560 MOV R0,#LSTMEN ;SAVE IT IN "LSTMEN"
2146 003454 163737 003330 001560 SUB #PROT,#LSTMEN ;SET PROTECTION
2147 003462 012737 057740 001556 MOV #ENDTAG+2,#FRSTMEN ;SET FIRST RELOCATION ADDRESS
2148
2149
2150 003470 012706 001200 MOV #KERSTK,SP ;SET STACK PTR
2151 003474 005037 001200 CLR #SPASS ;CLEAR PASS COUNT
2152 003500 105037 001545 CLRB #MMON ;SET MEM MGMT ON IND=NOT ON
2153 003504 012737 000600 001562 MOV #600,#NEXPAR ;SET FIRST 'PAR' VALUE
2154 003512 005737 003330 TST #PROT
2155 003516 001403 BEQ 1$
2156 003520 012737 001600 001562 MOV #1600,#NEXPAR
2157 003526 1$:
2158 003526 012700 000027 MOV #27,R0 ;SET SOB COUNT
2159 003532 005001 CLR R1 ;SETUP INDEX
2160 003534 005061 001642 2$: CLR MTICKS(R1) ;CLEAR TABLES
    
```





```

2217                                     ;TESTED.
2218
2219                                     ;CLEAR PROGRAM INDICATORS
2220 004070 052777 000100 175150      BIS      #100,#STKS      ;SET IE BIT IN KEYBOARD STATUS REG
2221 004076 012737 054152 000060      MOV      #TKISR,#TKVEC  ;SETUP KEYBOARD VECTOR
2222 004104 012737 000200 000062      MOV      #PR4,#TKVEC+2
2223 004112 012737 054364 000064      MOV      #TPISR,#TPVEC
2224 004120 012737 000200 000066      MOV      #PR4,#TPVEC+2
2225 004126 005037 001530      CLR      #NOTYPE      ;CLEAR 'NO TYPING' INDICATOR
2226
2227                                     ;THE BELOW ROUTINE ASCERTAINS WHICH CP & CP OPTIONS THE PROGRAM IS RUN-
2228                                     ;NING ON AND SETS AN INDICATOR IN OPT.CP ACCORDINGLY.
2229 004132 012737 000006 000004      CPCHK:  MOV      #ERRVEC+2,#ERRVEC  ;SET UP ERROR TRAP TO RETURN
2230 004140 012737 000002 000006      MOV      #2,#ERRVEC+2      ;RTI OPCODE = 2
2231 004146 012737 000012 000010      MOV      #RESVEC+2,#RESVEC  ;AND ALSO RESERVED INST TRAP
2232 004154 012737 000002 000012      MOV      #2,#RESVEC+2
2233 004162 012702 160000      MOV      #160000,R2      ;SET PROCESSOR'S NON-OPTION BITS
2234                                     ;EXCEPT FOR LKOPT
2235 004166 000261      SEC
2236 004170 005737 177546      TST      #LKS      ;BRANCH IF NO KW11-L
2237 004174 103402      BCS      7$      ;OR IF IT'S NOT WORKING
2238 004176 052702 001000      BIS      #LKOPT,R2      ;SET OPTION INDICATOR
2239 004202 000261      7$: SEC
2240 004204 005777 175042      TST      #STPS      ;BRANCH IF NO CONSOLE TTY
2241 004210 103402      BCS      9$
2242 004212 052702 000400      BIS      #TTOPT,R2
2243 004216 005003      9$: CLR      R3
2244 004220 000261      SEC
2245 004222 005737 170000      TST      #UBEDB      ;IS UBE1 THERE?
2246 004226 103410      BCS      12$      ;BRANCH IF NO
2247 004230 105037 170006      CLRB    #UBECR1      ;IS THIS A TESTER OR EXERCISOR?
2248 004234 105737 170006      TSTB    #UBECR1
2249 004240 100045      BPL      15$      ;BRANCH IF TESTER
2250 004242 052702 000200      16$: BIS      #UBEOPT,R2      ;SET INDICATOR
2251 004246 000425      BR      17$
2252 004250 000261      12$: SEC
2253 004252 005737 170020      TST      #UBEDB+20      ;IS UBE2 THERE?
2254 004256 103403      BCS      13$      ;BRANCH IF NO
2255 004260 012703 000020      MOV      #20,R3      ;SET OFFSET IN R3
2256 004264 000766      BR      16$
2257 004266 000261      13$: SEC
2258 004270 005737 170040      TST      #UBEDB+40      ;IS UBE3 THERE?
2259 004274 103403      BCS      14$      ;BRANCH IF NO
2260 004276 012703 000040      MOV      #40,R3      ;PUT OFFSET IN R3
2261 004302 000757      BR      16$
2262 004304 000261      14$: SEC
2263 004306 005737 170060      TST      #UBEDB+60      ;IS UBE4 THERE?
2264 004312 103420      BCS      15$      ;BRANCH IF NO
2265 004314 012703 000060      MOV      #60,R3      ;PUT OFFSET IN R3
2266 004320 000750      BR      16$
2267 004322 005227 177777      17$: INC      #-1
2268 004326 001012      BNE      15$
2269 004330 012704 002304      MOV      #UBETBL,R4      ;GET ADDRESS OF UBE TABLE
2270 004334 012705 000005      MOV      #5,R5      ;SET SOB COUNT
2271 004340 060324      18$: ADD      R3,(R4)+      ;ADJUST UBE TABLE ENTRIES
2272 004342 077502      SOB      R5,18$      ;CONTINUE
    
```

```

2273 004344 006003 ROR R3
2274 004346 006003 ROR R3 ;ADJUST OFFSET FOR UBE VECTOR
2275 004350 060324 ADD R3,(R4)+ ;ADJUST UBEVEC ENTRY
2276 004352 060314 ADD R3,(R4) ;ADJUST UBEVEC PSM ENTRY
2277 004354 005003 15$: CLR R3 ;INIT R3
2278 004356 000261 SEC
2279 004360 005777 175736 TST #MBTTBL ;IS MASS BUS TESTER THERE?
2280 004364 103403 BCS 20$ ;BRANCH IF NO
2281 004366 052702 002000 21$: BIS #MBOPT,R2 ;SET OPTION AVAILABLE
2282 004372 000422 BR 24$
2283 004374 005777 175752 20$: TST #MBTN2 ;IS MBT2 THERE?
2284 004400 103403 BCS 22$ ;BRANCH IF NO
2285 004402 012703 000100 MOV #100,R3 ;SETUP R3
2286 004406 000767 BR 21$
2287 004410 005777 175740 22$: TST #MBTN3 ;IS MBT3 THERE?
2288 004414 103403 BCS 23$ ;BRANCH IF NO
2289 004416 012703 000200 MOV #200,R3
2290 004422 000761 BR 21$
2291 004424 005777 175726 23$: TST #MBTN4 ;IS MBT4 THERE?
2292 004430 103427 BCS 30$ ;BRANCH IF NO
2293 004432 012703 000300 MOV #300,R3
2294 004436 000753 BR 21$
2295 004440 005227 177777 24$: INC #1
2296 004444 001021 BNE 30$
2297 004446 012704 002322 MOV #MBTTBL,R4 ;GET ADDRESS OF MBT TABLE
2298 004452 012705 000011 MOV #11,R5 ;SET SOB COUNT
2299 004456 060324 25$: ADD R3,(R4)+ ;ADJUST MBT TABLE
2300 004460 077502 SOB R5,25$ ;CONTINUE
2301 004462 060337 002350 ADD R3,#MBTTBL+26 ;ADJUST DRIVE TYPE ADDRESS
2302 004466 112777 000007 175640 MOVB #7,#MBTTBL+12
2303 004474 122777 000040 175646 CMPB #40,#MBTTBL+26 ;IS THIS REALLY A MBT?
2304 004502 001402 BEQ 30$ ;BRANCH IF YES
2305 004504 042702 002000 BIC #MBOPT,R2 ;CLEAR OPTION AVAILABLE BIT
2306 004510 012737 055064 000004 30$: MOV #ERRPT,#ERRVEC ;RESTORE ERROR TRAP
2307 004516 012737 054774 000010 MOV #RESERR,#RESVEC ;AND ALSO RESERVED INST TRAP
2308 004524 010237 001532 MOV R2,#OPT.CP ;LOAD INDICATOR
2309 .SBTTL TYPE PROGRAM NAME
2310 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
2311 004530 005227 177777 INC #1 ;FIRST TIME?
2312 004534 001037 BNE 64$ ;BRANCH IF NO
2313 004536 022737 041230 000042 CMP #SENDAD,#42 ;ACT-11?
2314 004544 001433 BEQ 64$ ;BRANCH IF YES
2315 004546 104401 004554 TYPE ,65$ ;TYPE ASCIZ STRING
2316 004552 000430 BR 64$ ;GET OVER THE ASCIZ
2317 65$: .ASCIZ <CRLF>"MAINDEC-11-DQKDC-A...PDP 11/6X CPU EXERCISER"<CRLF>
2318 004634 64$:
2319 *****
2320 .SBTTL SYSTEM SIZER
2321 THIS ROUTINE DETERMINES WHAT DRIVES ARE AVAILABLE ON
2322 THE FOLLOWING DEVICES: RK05, RPO3, RPO4, AND RSO4. THE
2323 INFORMATION IS STORED IN THE TABLE "SYSSIZE" IN THE FOLLOWING FORMAT:
2324 A. EACH DEVICE IS ASSIGNED A WORD
2325 B. THE LOW BYTE OF THIS WORD INDICATES WHICH DRIVES ARE AVAILABLE
2326 C. THE HIGH BYTE INDICATES WHICH DRIVES HAVE BEEN USED
2327 BY THE RELOCATION ROUTINE.
2328 *****

```



K05

```

2329 004634 012737 004746 000004 SIZE: MOV      #215, @#ERRVEC      ; SETUP TIMEOUT VECTOR
2330 004642 005037 001304          CLR      @#STMPD      ; ENSURE STMPD CLEAR
2331 004646 005000          CLR      R0          ; USED TO SET THE UNIT AVAIL BITS
2332 004650 012701 000010          MOV      #10, R1     ; SOB COUNT
2333 004654 013777 001304 175326 9S:  MOV      @#STMPD, @#RKA  ; SET UNIT NUMBER
2334 004662 012777 000015 175312  MOV      #15, @#RKS   ; SEND DRIVE RESET
2335 004670 032777 000200 175302  BIT      @#BIT7, @#RKR ; NON EXISTANT DISK?
2336 004676 001011          BNE      7S          ; BRANCH IF YES
2337 004700 017702 175272          MOV      @#RKS, R2   ; GET DRIVE STATUS
2338 004704 042702 177537          BIC      #177537, R2 ; GET BITS 5 & 7 ONLY
2339 004710 022702 000200          CMP      #200, R2    ; IS DRIVE READY?
2340 004714 001002          BNE      7S          ; BRANCH IF NO
2341 004716 052700 000400          BIS      @#BIT8, R0  ; SET UNIT AVAILABLE
2342 004722 006000          ROR      R0          ;
2343 004724 012777 000001 175250 7S:  MOV      #1, @#RKS   ; CLEAR THE ERRORS
2344 004732 062737 020000 001304  ADD      #20000, @#STMPD ; SELECT NEXT UNIT
2345 004740 077133          SOB      R1, 9S     ; CONTINUE
2346 004742 110037 001650          MOVB     R0, @#SYSSIZE+2 ; STORE IN TABLE
2347
2348 ; *****
2349 ; THIS CODE DETERMINES IF THERE IS AN RPO3 OR AN RPO4 OR BOTH.
2350 ; IF BOTH ARE ON THE SYSTEM, THE OPERATOR MUST CHANGE THE RPO4
2351 ; ADDRESSES IN THE TABLE IN "COMMON TAGS" AND "NOP" THE BRANCH
2352 ; AT "100S".
2353 ; *****
2354 004746 012737 005206 000004 21S: MOV      #115, @#ERRVEC ; SET THE ERROR VECTOR
2355 004754 005737 176710          TST      @#176710   ; IS THERE AN RP ON THE SYSTEM?
2356 ; STAY HERE IF YES
2357 004760 012737 004774 000004          MOV      #15, @#ERRVEC ;
2358 004766 005777 175224          TST      @#R4CS1   ; IS THERE AN RPO4 ON SYSTEM?
2359 004772 000441          BR      100S       ; BRANCH IF YES
2360 ; *****
2361 ; *****
2362 ; *****
2363 004774 012737 005076 000004 1S:  MOV      #105, @#ERRVEC ; SETUP TIMEOUT VEC FOR RPO3 TEST
2364 005002 012737 000001 001304  MOV      #1, @#STMPD ; SETUP TEMPO
2365 005010 005000          CLR      R0          ; USED TO SET UNIT AVAILABLE BITS
2366 005012 012701 000010          MOV      #10, R1    ; SOB COUNT
2367 005016 013777 001304 175134 3S:  MOV      @#STMPD, @#R3CS ; SET FUNCTION IDLE WITH UNIT NO
2368 005024 005777 175130          TST      @#R3CS    ; WAS THERE AN ERROR?
2369 005030 100006          BPL      6S          ; BRANCH IF NO
2370 005032 006000          ROR      R0          ; UNIT NOT AVAILABLE
2371 005034 062737 000400 001304 4S:  ADD      #400, @#STMPD ; SELECT NEXT UNIT
2372 005042 077113          SOB      R1, 3S     ; CONTINUE
2373 005044 000412          BR      5S          ;
2374 005046 017702 175102          MOV      @#R3DS, R2 ; GET STATUS REGISTER
2375 005052 042702 136377          BIC      #136377, R2 ; GET BITS 14, 9 & 8 ONLY
2376 005056 022702 041400          CMP      #41400, R2 ; IS DRIVE READY?
2377 005062 001363          BNE      4S          ; BRANCH IF NO
2378 005064 052700 000400          BIS      @#BIT8, R0 ; SET DRIVE AVAILABLE BIT
2379 005070 000760          BR      4S          ; CONTINUE
2380 005072 110037 001646          MOVB     R0, @#SYSSIZE ; STORE IN TABLE
2381
2382 ; *****
2383 005076 012737 005206 000004 10S: MOV      #115, @#ERRVEC ; SETUP ERROR VEC FOR RPO4 TEST
2384 005104 005037 001304          CLR      @#STMPD

```



```

2385 005110 005000          CLR      RO          ;UNIT AVAILABLE WORD
2386 005112 012701 000010    MOV      #10,R1      ;SOB COUNT
2387 005116 113777 001304 175104 14S:  MOVB    @#STMP0,@RP4CS2 ;SET UNIT NUMBER
2388 005124 012777 000021 175064    MOV      #21,@RP4CS1 ;TRY READ-IN-PRESET
2389 005132 032777 010000 175070    BIT      #BIT12,@RP4CS2 ;NON EXISTANT DRIVE?
2390 005140 001011          BNE      12S         ;BRANCH IF YES
2391 005142 017702 175066    MOV      @RP4DS,R2   ;GET DRIVE STATUS
2392 005146 042702 163277    BIC      #163277,R2  ;GET BITS 12, 11, 8, & 6 ONLY
2393 005152 022702 010500    CMP      #10500,R2   ;IS DRIVE READY?
2394 005156 001002          BNE      12S         ;BRANCH IF NO
2395 005160 052700 000400    BIS      #BIT8,RO    ;SET UNIT AVAILABLE
2396 005164 006000          ROR      RO
2397 005166 052777 000040 175034 12S:  BIS      #BITS,@RP4CS2 ;CLEAR ERROR BITS
2398 005174 005237 001304    INC      @#STMP0     ;SELECT NEXT DRIVE
2399 005200 077132          SOB      R1,14S     ;CONTINUE
2400 005202 110037 001656    MOVB    RO,@#SYSSIZE+10 ;STORE IN TABLE
2401
2402
2403 005206 012737 005316 000004 11S:  MOV      #15S,@#ERRVEC ;SETUP ERROR VEC FOR RSO4 TEST
2404 005214 005037 001304          CLR      @#STMP0
2405 005220 005000          CLR      RO
2406 005222 012701 000010    MOV      #10,R1      ;SOB COUNT
2407 005226 113777 001304 175034 18S:  MOVB    @#STMP0,@RSCS2 ;SET UNIT NUMBER
2408 005234 012777 000001 175014    MOV      #1,@RSCS1  ;TRY NOP OPERATION
2409 005242 032777 010000 175020    BIT      #BIT12,@RSCS2 ;NON EXISTANT DRIVE?
2410 005250 001011          BNE      16S         ;BRANCH IF YES
2411 005252 017702 175016    MOV      @RSDS,R2   ;GET DRIVE STATUS
2412 005256 042702 163577    BIC      #163577,R2  ;GET BITS 12, 11, & 7 ONLY
2413 005262 022702 010200    CMP      #10200,R2  ;IS DRIVE READY?
2414 005266 001002          BNE      16S         ;BRANCH IF NO
2415 005270 052700 000400    BIS      #BIT8,RO    ;SET DRIVE AVAILABLE BIT
2416 005274 006000          ROR      RO
2417 005276 052777 000040 174764 16S:  BIS      #BITS,@RSCS2 ;CLEAR ANY ERROR BITS
2418 005304 005237 001304    INC      @#STMP0     ;SELECT NEXT UNIT
2419 005310 077132          SOB      R1,18S     ;CONTINUE
2420 005312 110037 001660    MOVB    RO,@#SYSSIZE+12 ;STORE IN TABLE
2421
2422
2423 005316 122737 000002 000041 15S:  CMPB    #2,@#41      ;RK?
2424 005324 001004          BNE      19S         ;BRANCH IF NO
2425 005326 042737 000001 001650    BIC      #BIT0,@#SYSSIZE+2 ;MAKE UNIT ZERO NOT AVAILABLE
2426 005334 000420          BR      20S
2427 005336 113700 000041 19S:  MOVB    @#41,RO      ;GET LOCATION 41
2428 005342 042700 177770    BIC      #177770,RO  ;GET LEAST SIG 3 BITS
2429 005346 000241          CLC
2430 005350 006100          ROL      RO          ;ENSURE C CLEAR
2431 005352 122700 000002          CMPB    #2,RO
2432 005356 002404          BLT     40S         ;ADJUST
2433 005360 042737 000001 001646    BIC      #BIT0,@#SYSSIZE ;BRANCH IF NO
2434 005366 000403          BR      20S
2435 005370 042760 000001 001652 40S:  BIC      #BIT0,SYSSIZE+4(RO)
2436 005376 005737 001546 20S:  TST     @#QV        ;ACT11?
2437 005402 001052          BNE     LOOP        ;BRANCH IF YES
2438 005404 105737 003325    TSTB   @#XXDPC      ;XXDP CHAIN MODE?
2439 005410 001047          BNE     LOOP        ;BRANCH IF YES
2440 005412 105737 001200    TSTB   @#SPASS      ;FIRST PASS?

```

M05

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58 SYSTEM SIZER

MACY11 27(1006) 07-FEB-77 10:08 PAGE 64

2441	005416	001044			BNE	LOOP	::BRANCH IF NO
2442	005420	005227	177777		INC	#-1	
2443	005424	001041			BNE	LOOP	::BRANCH IF NOT FIRST TIME
2444	005426	032777	000200	173606	BIT	#SW7,#SWR	::INHIBIT SIZE TYPEOUT?
2445	005434	001035			BNE	LOOP	::BRANCH IF YES
2446	005436	032737	000400	001532	BIT	#BIT8,#OPT.CP	::TTY AVAILABLE?
2447	005444	001431			BEQ	LOOP	::BRANCH IF NO TTY
2448	005446	004767	043576		JSR	PC,TYPSIZ	::GO TYPE SYSTEM SIZE
2449	005452	104401	005460		TYPE	,65\$	::TYPE ASCIZ STRING
2450	005456	000417			BR	,64\$	::GET OVER THE ASCIZ
2451					BR	,65\$:	::.ASCIZ
2452	005516				BR	,64\$:	::/TYPE A CHARACTER TO CONTINUE/<CR LF>
2453	005516	005037	177776		CLR	#PSW	
2454	005522	000001			WAIT		
2455	005524	000137	004634		JMP	#SIZE	::GO CHECK SYSTEM AGAIN



;PROGRAM RESTARTS HERE AFTER RELOCATION ABOVE 28K IS COMPLETE.

;INITIALIZE TRAP VECTORS

```

LOOP:  MOV      #USESTK,SP          ;SET THE STACK...WILL BE DIFFERENT
                                           ;THAN KERN STACK WHEN IN OUTER MODE
        MOV      #ERRVEC,RO
        MOV      @#PSW,R1          ;GET CURRENT PSW
        MOV      #ERPT,(RO)+       ;SET ERROR VEC
        BIS      #PR7,R1          ;SET PRIORITY 7 IN CURRENT PSW
        BIC      #BIT4,R1
        MOV      R1,(RO)+
        MOV      #RESERR,(RO)+     ;SET RESERVED INST TRAP VECTOR
        MOV      R1,(RO)+
        MOV      #SATRN,(RO)+     ;SET T BIT VEC
        CLR      (RO)+             ;SET TBIT VEC+2
        TST      (RO)+             ;BUMP RO TO SCOPE VEC+2
        CLR      (RO)+             ;SET SCOPE VEC+2
        ADD      #6,RO             ;SET RO TO ERROR TRAP VEC
        MOV      #PR7,(RO)+       ;SET ERROR VEC
        TST      (RO)+
        MOV      #PR7,(RO)+       ;SET TRAP VEC+2
        MOV      #.PARSRV,@#CACHVEC ;SET PARITY ERROR VECTOR
        MOV      R1,@#CACHVEC+2
        MOV      #KTABRT,@#MMVEC   ;SET MEM. MGMT. ABORT VECTOR.
        MOV      R1,@#MMVEC+2
        CLR      @DISPLAY
        BIC      #PR7,@#PSW

```

```

;*****
;TEST 1 MEMORY VERIFICATION TEST
;*****

```

```

TST1:  MOV      #1,STIMES          ;;DO 1 ITERATION
        SCOPE
        MOVB    #1,@#STSTNM
        .SBTTL  START OF SECTION 0
        .0000000000000000 FIRST ADDRESS TO BE RELOCATED 00000000
RELO:  MOVB    #STN-1,@#STSTNM
        MOV     PC,RO              ;GET PC
        TST    -(RO)              ;RO CONTAINS THE ADDRESS OF RELO
        MOV    RO,@#FRSTAD         ;SAVE
        MOV    PC,RO              ;GET CURRENT PC
        SUB    #.,RO              ;SUBTRACT RELOCATION FACTOR
        MOV    RO,@#FACTOR         ;SAVE RELOCATION FACTOR
        MOV    PC,@#SLPERR        ;SET LOOP ADDRESS
        ADD    #26,@#SLPERR       ;ADJUST
        MOV    @#SLPERR,@#SLPADR
        TSTB   @#NEXEC            ;BR IF TEST CODE TO BE EXECUTED
        BEQ    +6
        JMP    RELO

```

;MEMORY AND DISK (IF SELECTED) VERIFICATION TEST.

```

JMP    IS
.WORD  -1,-1,-1,-1,0,0,0,0
.WORD  -1,-1,-1,-1,0,0,0,0

```

2456					
2457					
2458	005530	012706	000700		
2459					
2460	005534	012700	000004		
2461	005540	013701	177776		
2462	005544	012720	055064		
2463	005550	052701	000340		
2464	005554	042701	000020		
2465	005560	010120			
2466	005562	012720	054774		
2467	005566	010120			
2468	005570	012720	001534		
2469	005574	005020			
2470	005576	005720			
2471	005600	005020			
2472	005602	062700	000006		
2473	005606	012720	000340		
2474	005612	005720			
2475	005614	012720	000340		
2476	005620	012737	054414	000114	
2477	005626	010137	000116		
2478	005632	012737	054662	000250	
2479	005640	010137	000252		
2480	005644	005077	173374		
2481	005650	042737	000340	177776	
2482					
2483					
2484					
2485	005656				
2486	005656	012767	000001	173440	
2487	005664	000004			
2488	005666	112737	000001	001202	
2489					
2490					
2491					
2492	005674	112737	000001	001202	
2493	005702	010700			
2494	005704	005740			
2495	005706	010037	001554		
2496	005712	010700			
2497	005714	162700	005714		
2498	005720	010037	001550		
2499	005724	010737	001212		
2500	005730	062737	000026	001212	
2501	005736	013737	001212	001210	
2502	005744	105737	001544		
2503	005750	001402			
2504	005752	000167	000720		
2505					
2506	005756	000167	000714		
2507	005762	177777	177777	177777	
2508	005770	177777	000000	000000	
2509	005776	000000	000000		
2510	006002	177777	177777	177777	
2511	006010	177777	000000	000000	



2512	006016	000000	000000		
2513	006022	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2514	006030	177777	000000	000000	
2515	006036	000000	000000		
2516	006042	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2517	006050	177777	000000	000000	
2518	006056	000000	000000		
2519	006062	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2520	006070	177777	000000	000000	
2521	006076	000000	000000		
2522	006102	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2523	006110	177777	000000	000000	
2524	006116	000000	000000		
2525	006122	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2526	006130	177777	000000	000000	
2527	006136	000000	000000		
2528	006142	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2529	006150	177777	000000	000000	
2530	006156	000000	000000		
2531	006162	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2532	006170	177777	000000	000000	
2533	006176	000000	000000		
2534	006202	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2535	006210	177777	000000	000000	
2536	006216	000000	000000		
2537	006222	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2538	006230	177777	000000	000000	
2539	006236	000000	000000		
2540	006242	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2541	006250	177777	000000	000000	
2542	006256	000000	000000		
2543	006262	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2544	006270	177777	000000	000000	
2545	006276	000000	000000		
2546	006302	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2547	006310	177777	000000	000000	
2548	006316	000000	000000		
2549	006322	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2550	006330	177777	000000	000000	
2551	006336	000000	000000		
2552	006342	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2553	006350	177777	000000	000000	
2554	006356	000000	000000		
2555	006362	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2556	006370	177777	000000	000000	
2557	006376	000000	000000		
2558	006402	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2559	006410	177777	000000	000000	
2560	006416	000000	000000		
2561	006422	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2562	006430	177777	000000	000000	
2563	006436	000000	000000		
2564	006442	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0
2565	006450	177777	000000	000000	
2566	006456	000000	000000		
2567	006462	177777	177777	177777	.WORD -1,-1,-1,-1,0,0,0,0





```

2624 007014 000257 ; CCC ;CC'S=0000
2625 007016 103407 ; BCS ;SAME AS BLO
2626 007016 103407 ; BCS CCD
2627 007020 102406 ; BVS CCD
2628 007022 001405 ; BEQ CCD
2629 007024 100404 ; BMI CCD
2630 007026 002403 ; BLT CCD
2631 007030 003402 ; BLE CCD
2632 007032 101401 ; BLOS CCD
2633 007034 101001 ; BHI .+4
2634 007036 104000 ; HLT ;ONE OF THE ABOVE BRANCHES FAILED
2635
2636 ;CONTINUE
2637 007040 000270 ; SEN ;CC'S=1000
2638 007042 100003 ; BPL CC1
2639 007044 002002 ; BGE CC1
2640 007046 003001 ; BGT CC1
2641 007050 002401 ; BLT .+4
2642 007052 104000 ; HLT ;ONE OF THE ABOVE BRANCHES FAILED
2643
2644 ;CONTINUE
2645 007054 000262 ; SEV ;CC'S=1010
2646 007056 102003 ; BVC CC2
2647 007060 002402 ; BLT CC2
2648 007062 003401 ; BLE CC2
2649 007064 002001 ; BGE .+4
2650 007066 104000 ; HLT ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
2651
2652 ;CONTINUE
2653 007070 000261 ; SEC ;CC'S=1011
2654 007072 103002 ; BCC CC3
2655 007074 101001 ; BHI CC3
2656 007076 003001 ; BGT .+4
2657 007100 104000 ; HLT ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
2658
2659 ;CONTINUE
2660 007102 000264 ; SEZ ;CC'S=1111
2661 007104 001003 ; BNE CC4
2662 007106 003002 ; BGT CC4
2663 007110 101001 ; BHI CC4
2664 007112 003401 ; BLE .+4
2665 007114 104000 ; HLT ;ERROR! ONE OF THE ABOVE BRANCHES FAILED
2666
2667 ;*****
2668 ;*TEST 3 TEST UNIARY CONDITION CODES
2669 ;*****
2670 ;TST3:
2671 007116 000004 SCOPE
2672 007120 112737 000003 001202 MOVB #3,#STSTNM
2673 ;CLR RO
2674 007126 000277 SCC
2675 007130 000244 CLZ
2676 007132 005000 CLR RO ;RO=0,CC'S=0100
2677 007134 103404 BCS CLRO
2678 007136 102403 BVS CLRO
2679 007140 001002 BNE CLRO
2679 007142 100401 BMI CLRO

```



# E06

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCA.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 69  
T3 TEST UNIARY CONDITION CODES

2680	007144	003401		BLE	.+4	
2681	007146	104000	CLRD:	HLT		;ERROR! INCORRECT CC'S AFTER CLR
2682						
2683	007150	000277		SCC		
2684	007152	000244		CLZ		
2685	007154	005700		TST	RO	;RO=0,CC'S=0100
2686	007156	103404		BCS	TSTO	
2687	007160	102403		BVS	TSTO	
2688	007162	001002		BNE	TSTO	
2689	007164	100401		BMI	TSTO	
2690	007166	101401		BLOS	.+4	
2691	007170	104000	TSTO:	HLT		;ERROR! INCORRECT CC'S AFTER TST
2692						
2693	007172	000257		CCC		
2694	007174	000266		+SEZ!SEV		
2695	007176	005100		COM	RO	;RO=-1,CC'S=1001
2696	007200	103004		BCC	COMO	
2697	007202	102403		BVS	COMO	
2698	007204	001402		BEQ	COMO	
2699	007206	100001		BPL	COMO	
2700	007210	002401		BLT	.+4	
2701	007212	104000	COMO:	HLT		;ERROR! INCORRECT CC'S AFTER COM
2702						
2703	007214	000261		SEC		
2704	007216	005500		ADC	RO	;RO=000000,CC'S=0101
2705	007220	103003		BCC	ADCO	
2706	007222	102402		BVS	ADCO	
2707	007224	001001		BNE	ADCO	
2708	007226	002001		BGE	.+4	
2709	007230	104000	ADCO:	HLT		;ERROR! INCORRECT CC'S AFTER ADC
2710						
2711	007232	000261		SEC		
2712	007234	006000		ROR	RO	;RO=100000,CC'S=1010
2713	007236	103404		BCS	RORO	
2714	007240	102003		BVC	RORO	
2715	007242	001402		BEQ	RORO	
2716	007244	100001		BPL	RORO	
2717	007246	003001		BGT	.+4	
2718	007250	104000	RORO:	HLT		;ERROR! INCORRECT CC'S AFTER ROR
2719	007252	000277		SCC		
2720	007254	000242		CLV		
2721	007256	005300		DEC	RO	;RO=077777,CC'S=0011
2722	007260	101004		BCC	DECO	
2723	007262	102003		BVC	DECO	
2724	007264	001402		BEQ	DECO	
2725	007266	100401		BMI	DECO	
2726	007270	003401		BLE	.+4	
2727	007272	104000	DECO:	HLT		;ERROR! INCORRECT CC'S AFTER DEC
2728						
2729	007274	000257		CCC		
2730	007276	005200		INC	RO	;RO=100000,CC'S=1010
2731	007300	103404		BCS	INCO	
2732	007302	102003		BVC	INCO	
2733	007304	001402		BEQ	INCO	
2734	007306	100001		BPL	INCO	
2735	007310	003001		BGT	.+4	

T3 TEST UNIARY CONDITION CODES

```

2736 007312 104000      INCO:  HLT                ;ERROR! INCORRECT CC'S AFTER INC
2737
2738 007314 000277      SCC
2739 007316 000242      CLV
2740 007320 005400      NEG      RO                ;RO=100000,CC'S=1011
2741 007322 103003      BCC      NEGO
2742 007324 102002      BVC      NEGO
2743 007326 001401      BEQ      NEGO
2744 007330 002001      BGE      .+4
2745 007332 104000      NEGO:  HLT                ;ERROR! INCORRECT CC'S AFTER NEG
2746
2747 007334 000261      SEC
2748 007336 006300      ASL      RO                ;RO=000000,CC'S=0111
2749 007340 103004      BCC      ASLO
2750 007342 102003      BVC      ASLO
2751 007344 001002      BNE      ASLO
2752 007346 100401      BMI      ASLO
2753 007350 101401      BLOS     .+4
2754 007352 104000      ASLO:  HLT                ;ERROR! INCORRECT CC'S AFTER ASL
2755
2756 007354 006100      ROL      RO                ;RO=000001,CC'S=0000
2757 007356 103402      BCS      ROLO
2758 007360 003401      BLE      ROLO
2759 007362 002001      BGE      .+4
2760 007364 104000      ROLO:  HLT                ;ERROR! INCORRECT CC'S AFTER ROL
2761
2762 007366 006200      ASR      RO                ;RO=000000,CC'S=0111
2763 007370 103003      BCC      ASRO
2764 007372 102002      BVC      ASRO
2765 007374 001001      BNE      ASRO
2766 007376 002401      BLT      .+4
2767 007400 104000      ASRO:  HLT                ;ERROR! INCORRECT CC'S AFTER ASR
2768
2769 007402 000277      SCC
2770 007404 005600      SBC      RO                ;RO=-1,CC'S=1001
2771 007406 103002      BCC      SBCO
2772 007410 102401      BVS      SBCO
2773 007412 003401      BLE      .+4
2774 007414 104000      SBCO:  HLT                ;ERROR! INCORRECT CC'S AFTER SBC
2775
2776 007416 005400      NEG      RO                ;RO=000001,CC'S=00001
2777 007420 000300      SWAB     RO                ;RO=000400 ,CC'S=0100
2778 007422 103403      BCS      SWABO
2779 007424 102402      BVS      SWABO
2780 007426 001001      BNE      SWABO
2781 007430 002001      BGE      .+4
2782 007432 104000      SWABO: HLT                ;ERROR! INCORRECT CC'S AFTER SWAB
2783
2784
2785
2786 007434
2787 007434 000004      SCOPE
2788 007436 112737 000004 001202      MOV      #4,0#STSTNM
2789 007444 012737 000005 001324      MOV      #5,0#STIMES

```

```

2790 007452 005000          CLR      R0
2791 007454 000277          SCC
2792 007456 006100          ROL      R0          ;R0=1
2793 007460 010002          MOV      R0,R2
2794 007462 006302          ASL      R2          ;R2=2
2795 007464 010203          MOV      R2,R3
2796 007466 006303          ASL      R3          ;R3=4
2797 007470 010304          MOV      R3,R4
2798 007472 006304          ASL      R4          ;R4=10
2799 007474 010405          MOV      R4,R5
2800 007476 006305          ASL      R5          ;R5=20
2801 007500 010546          MOV      R5,-(SP)   ;SET BITS SET IN REGISTERS
2802 007502 050416          BIS      R4,(SP)   ;INTO STACK ADDRESS
2803 007504 050316          BIS      R3,(SP)
2804 007506 050216          BIS      R2,(SP)
2805 007510 050016          BIS      R0,(SP)
2806 007512 022726 000037  CMP      #37,(SP)+
2807 007516 001401          BEQ      .+4        ;WERE SET
2808 007520 104000          HLT
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835

```

000037

000377

000020

100000

```

;CHECK THAT ALL BITS CAN BE SET & CLEARED IN ALL REGISTERS
CCC
1$: MOVB    #377,R0      ;SET ALL BITS (MOVB EXTENDS SIGN)
   ROL     R0           ;ROTATE A 0 THROUGH ALL BIT
   BCS    1$           ;POSITIONS
   INC    R0           ;FINAL RESULT IS -1
   BEQ    .+4
   HLT
   ;ERROR!

2$: MOV     #16.,R0     ;SET SHIFT COUNT
   CLR    R2
   SEC
   ROR    R2           ;ROTATE 1 THROUGH ALL BIT POSITS
   DEC    R0           ;DECREMENT SHIFT COUNT
   BNE    2$
   COM    R2           ;R2 SHOULD CONTAIN -1
   BEQ    .+4
   HLT
   ;ERROR! CHECK R2 SHOULD = 0

3$: MOV     #100000,R3
   ASR    R3           ;EXTEND 1 BIT THROUGH ALL POSITIONS
   BCC    3$
   INC    R3
   BEQ    .+4
   HLT
   ;ERROR!

```



```

2836
2837 007604 112704 177401      MOVB    #177401,R4      ;R4=1
2838 007610 060404      4S:    ADD     R4,R4      ;HAS THE AFFECT OF SHIFTING A BIT
2839 007612 103376      BCC     4S              ;THROUGH ALL POSITIONS
2840 007614 005704      TST     R4              ;RESULT SHOULD BE 0
2841 007616 001401      BEQ     .+4
2842 007620 104000      HLT
2843
2844 007622 012705 000001      MOV     #1,R5
2845 007626 006305      5S:    ASL     R5
2846 007630 102376      BVC     5S
2847 007632 006305      ASL     R5
2848 007634 103002      BCC     6S
2849 007636 005705      TST     R5
2850 007640 001401      BEQ     .+4
2851 007642 104000      6S:    HLT
2852
2853      ;CHECK REGISTER VOLITILITY
2854 007644 005002      CLR     R2
2855 007646 005102      COM     R2              ;R2=-1
2856 007650 010203      MOV     R2,R3
2857 007652 000257      CCC
2858 007654 006002      ROR     R2              ;R2=LOOP COUNT
2859 007656 006202      ASR     R2
2860 007660 010304      7S:    MOV     R3,R4
2861 007662 005302      DEC     R2              ;DECREMENT LOOP COUNT
2862 007664 001375      BNE     7S
2863 007666 005203      INC     R3              ;CHECK R3
2864 007670 001002      BNE     8S
2865 007672 005204      INC     R4              ;CHECK R4
2866 007674 001401      BEQ     .+4
2867 007676 104000      8S:    HLT
2868
2869      ;CHECK TRANSFER OF REGISTER DATA BETWEEN THE GS AND GD REGISTERS
2870 007700 032737 000020 177776  GSTST:  BIT     #20,#PSW      ;CHECK IF 'T' BIT IS SET
2871 007706 001050      BNE     7S              ;SKIP TEST IF 'T' BIT SET
2872 007710 010627      MOV     SP,(PC)+        ;SAVE STACK PTR
2873 007712 000000      1S:    .WORD 0              ;CONTAINS SAVED STACK PTR
2874 007714 010727      MOV     PC,(PC)+        ;LOAD DATA. THE CURRENT PC IS USED AS
2875 007716 000000      2S:    .WORD 0              ;DATA. IF THIS TEST FAILS 2S CON-
2876                                     ;TAINS THE DATA BEING USED.
2877 007720 005267 177772      INC     2S              ;MAKE ODD TO CHECK BIT 0
2878 007724 016700 177766      3S:    MOV     2S,R0          ;LOAD GD REGISTER 0
2879 007730 010001      MOV     R0,R1          ;TRANSFER GS REG 0 TO GD REG 1
2880 007732 010102      MOV     R1,R2          ;AND GS REG 1 TO GD REG 2
2881 007734 010203      MOV     R2,R3          ;ETC...
2882 007736 010304      MOV     R3,R4
2883 007740 010405      MOV     R4,R5
2884 007742 152737 000340 177776  BISB    #340,#PSW      ;SET PRIORITY LEVEL 7
2885 007750 010506      MOV     R5,SP          ;TRANSFER GS REG 5 TO GD STK PTR
2886 007752 010627      MOV     SP,(PC)+        ;TRANSFER GS STK PTR TO MEMORY
2887 007754 000000      4S:    .WORD 0              ;CONTAINS GS STACK PTR
2888 007756 016706 177730      MOV     1S,SP          ;RESTORE STK PTR NEEDED FOR HLT/SCOPE
2889 007762 142737 000340 177776  BICB    #340,#PSW      ;SET PRIORITY LEVEL 0
2890 007770 026700 177760      CMP     4S,R0          ;COMPARE GS/GD STACK WITH GS REG 0
2891 007774 001004      BNE     5S              ;BRANCH IF THEY WERE NOT =

```

```

2892 007776 006367 177714      ASL      2$      ;SHIFT TEST DATA UNTIL = 000000
2893 010002 001350      BNE      3$
2894 010004 000411      BR       6$
2895 010006 010046      SS:     MOV      R0,-(SP)    ;GET GS REG 0
2896 010010 010146      MOV      R1,-(SP)    ;ETC...
2897 010012 010246      MOV      R2,-(SP)
2898 010014 010346      MOV      R3,-(SP)
2899 010016 010446      MOV      R4,-(SP)
2900 010020 010546      MOV      R5,-(SP)
2901 010022 104000      HLT
2902                                     ;ERROR! DATA IN GS STK PTR NOT = GS REG 0
2903 010024 016706 177662      MOV      1$,SP      ;GS REG 0-GS REG 5 ARE ON THE STACK
2904 010030                                     ;RESTORE STACK PTR
2905 010030      6$:
2906                                     7$:
2907                                     ;*****
2908                                     ;*TEST 5      TEST UNIARY WORD INSTRUCTIONS USING ADDRESS MODE 1
2909                                     ;*****
2910 010030      TSTS:
2911 010030 000004      SCOPE
2912 010032 112737 000005 001202      MOV      #5,2#$STSTNM
2913 010040 012737 000005 001324      MOV      #5,2#$STIMES
2914 010046 000401      BR       +4
2915 010050 000000      .WORD   0      ;RESERVE ADDRESS FOR TESTS
2916 010052 010702      MOV      PC,R2
2917 010054 162702 000004      SUB      #4,R2      ;R2 POINTS TO RESERVED WORD
2918 010060 005012      CLR      (R2)      ;PRESET (R2)
2919 010062 000261      SEC
2920 010064 006012      ROR      (R2)      ;(R2)=100000,CC=1010
2921 010066 101402      BLOS    ROR1
2922 010070 100001      BPL     ROR1
2923 010072 002001      BGE     .+4
2924 010074 104000      ROR1:   HLT      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2925
2926 010076 000257      CCC
2927 010100 000261      SEC
2928 010102 005312      DEC      (R2)      ;(R2)=077777,CC=0011
2929 010104 103001      BCC     DEC1
2930 010106 003401      BLE     .+4
2931 010110 104000      DEC1:   HLT      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2932
2933 010112 000257      CCC
2934 010114 000261      SEC
2935 010116 005512      ADC      (R2)      ;(R2)=100000,CC=1010
2936 010120 103403      BCS     ADC1
2937 010122 102002      BVC     ADC1
2938 010124 100001      BPL     ADC1
2939 010126 001001      BNE     .+4
2940 010130 104000      ADC1:   HLT      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2941
2942 010132 006112      ROL      (R2)      ;(R2)=000000,CC=0111
2943 010134 103003      BCC     ROL1
2944 010136 102002      BVC     ROL1
2945 010140 001001      BNE     ROL1
2946 010142 100001      BPL     .+4
2947 010144 104000      ROL1:   HLT      ;ERROR! INCORRECT CC'S AS SHOWN ABOVE

```

# JOB

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 74  
TS TEST UNINARY WORD INSTRUCTIONS USING ADDRESS MODE 1

2948					
2949	010146	006112	ROL	(R2)	;(R2)=000001,CC=0000
2950	010150	101402	BLOS	ROL1A	;BRANCH IF C OR Z IS SET
2951	010152	102401	BVS	ROL1A	
2952	010154	100001	BPL	.+4	
2953	010156	104000	ROL1A:	HLT	
2954					
2955	010160	006212	ASR	(R2)	;(R2)=000000,CC=0111
2956	010162	103003	BCC	ASR1	
2957	010164	102002	BVC	ASR1	
2958	010166	001001	BNE	ASR1	
2959	010170	100001	BPL	.+4	
2960	010172	104000	ASR1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2961					
2962	010174	006012	ROR	(R2)	;(R2)=100000,CC=1010
2963	010176	103403	BCS	ROR1A	
2964	010200	102002	BVC	ROR1A	
2965	010202	001401	BEQ	ROR1A	
2966	010204	100401	BMI	.+4	
2967	010206	104000	ROR1A:	HLT	
2968					
2969	010210	000261	SEC		
2970	010212	005212	INC	(R2)	;(R2)=100001,CC=1001
2971	010214	103003	BCC	INC1	
2972	010216	102402	BVS	INC1	
2973	010220	001401	BEQ	INC1	
2974	010222	100401	BMI	.+4	
2975	010224	104000	INC1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2976					
2977	010226	005612	SBC	(R2)	;(R2)=100000,CC=1000
2978	010230	103403	BCS	SBC1	
2979	010232	102402	BVS	SBC1	
2980	010234	001401	BEQ	SBC1	
2981	010236	100401	BMI	.+4	
2982	010240	104000	SBC1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2983					
2984	010242	000261	SEC		
2985	010244	005612	SBC	(R2)	;(R2)=077777,CC=0010
2986	010246	103403	BCS	SBC1A	
2987	010250	102002	BVC	SBC1A	
2988	010252	001401	BEQ	SBC1A	
2989	010254	100001	BPL	.+4	
2990	010256	104000	SBC1A:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
2991					
2992	010260	000261	SEC		
2993	010262	005512	ADC	(R2)	;(R2)=100000,CC=1010
2994	010264	100401	BMI	.+4	
2995	010266	104000	HLT		
2996					
2997	010270	000261	SEC		
2998	010272	006312	ASL	(R2)	;(R2)=000000,CC=0111
2999	010274	103003	BCC	ASL1	
3000	010276	102002	BVC	ASL1	
3001	010300	001001	BNE	ASL1	
3002	010302	100001	BPL	.+4	
3003	010304	104000	ASL1:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE



```

3004
3005 010306 005112          COM      (R2)          ;(R2)=177777,CC=1001
3006 010310 103002          BCC     COM1
3007 010312 102401          BVS     COM1
3008 010314 100401          BMI     .+4
3009 010316 104000          COM1:  HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3010
3011 010320 000250          CLN
3012 010322 005712          TST     (R2)          ;(R2)=177777,CC=1000
3013 010324 103403          BCS     TEST1
3014 010326 102402          BVS     TEST1
3015 010330 100001          BPL     TEST1
3016 010332 001001          BNE     .+4
3017 010334 104000          TEST1: HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3018
3019 010336 000262          SEV
3020 010340 005412          NEG     (R2)          ;(R2)=000001,CC=0000
3021 010342 103002          BCC     NEG1
3022 010344 102401          BVS     NEG1
3023 010346 001001          BNE     .+4
3024 010350 104000          NEG1:  HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3025
3026 010352 005312          DEC     (R2)          ;(R2)=000000,CC=0101
3027 010354 103001          BCC     DEC1A
3028 010356 001401          BEQ     .+4
3029 010360 104000          DEC1A: HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3030
3031 *****
3032 ;#TEST 6 CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1
3033 *****
3034 010362 000004          TST6:  SCOPE
3035 010364 112737 000006 001202  MOVB    #6,#STSTNM
3036 010372 000401          BR      .+4          ;RESERVE A WORD
3037 010374 000000          .WORD  0          ;ADDRESS RESERVED FOR TESTS
3038 010376 010703          MOV     PC,R3
3039 010400 162703 000004  SUB     #4,R3          ;R3 POINTS TO EVEN BYTE OF WORD
3040 010404 010304          MOV     R3,R4          ;R4 POINTS TO ODD BYTE OF WORD
3041 010406 005204          INC     R4
3042 010410 005013          CLR     (R3)          ;PRESET DATA
3043
3044 010412 000261          1$:    SEC
3045 010414 105513          ADCB   (R3)          ;ADD CARRY TO EVEN BYTE
3046 010416 100402          BMI   2$          ;UNTIL EVEN BYTE BECOMES NEGATIVE
3047 010420 105214          INCB   (R4)          ;INCREMENT ODD BYTE
3048 010422 000773          BR     1$
3049 010424 102401          2$:    BVS     .+4          ;(R3)=077600=[0774][200],CC=1010
3050 010426 104000          HLT
3051 010430 000242          CLV
3052 010432 105214          INCB   (R4)          ;(R3)=100200=[1000][200],CC=1010
3053 010434 103402          BCS   INCB1
3054 010436 102001          BVC   INCB1
3055 010440 100401          BMI   .+4
3056 010442 104000          INCB1: HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3057
3058 010444 106114          ROLB   (R4)          ;(R3)=000200=[0000][200],CC=0111
3059 010446 103002          BCC   ROLB1

```

3060	010450	102001		BVC	ROLB1	
3061	010452	001401		BEQ	.+4	
3062	010454	104000	ROLB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3063						
3064	010456	105614		SBCB	(R4)	
3065	010460	103002		BCC	SBCB1	
3066	010462	102401		BVS	SBCB1	
3067	010464	100401		BMI	.+4	
3068	010466	104000	SBCB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3069						
3070	010470	106313		ASLB	(R3)	
3071	010472	103002		BCC	ASLB1	
3072	010474	102001		BVC	ASLB1	
3073	010476	001401		BEQ	.+4	
3074	010500	104000	ASLB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3075						
3076	010502	105413		NEGB	(R3)	
3077	010504	103402		BCS	NEGB1	
3078	010506	102401		BVS	NEGB1	
3079	010510	001401		BEQ	.+4	
3080	010512	104000	NEGB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3081						
3082	010514	000277		SCC		
3083	010516	105313		DECB	(R3)	
3084	010520	103002		BCC	DECB1	
3085	010522	102401		BVS	DECB1	
3086	010524	001001		BNE	.+4	
3087	010526	104000	DECB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3088						
3089	010530	000241		CLC		
3090	010532	106013		RORB	(R3)	
3091	010534	103002		BCC	RORB1	
3092	010536	102001		BVC	RORB1	
3093	010540	100001		BPL	.+4	
3094	010542	104000	RORB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3095						
3096	010544	000241		CLC		
3097	010546	105114		COMB	(R4)	
3098	010550	103002		BCC	COMB1	
3099	010552	102401		BVS	COMB1	
3100	010554	001401		BEQ	.+4	
3101	010556	104000	COMB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3102						
3103	010560	106213	1S:	ASRB	(R3)	;SHIFT EVEN BYTE UNTIL V CLEARS
3104	010562	102002		BVC	2S	
3105	010564	105514		ADCB	(R4)	;AND ADD CARRY TO ODD BYTE
3106	010566	000774		BR	1S	
3107	010570	103401	2S:	BCS	ASRB1	
3108	010572	001401		BEQ	.+4	
3109	010574	104000	ASRB1:	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3110						
3111	010576	106214		ASRB	(R4)	
3112	010600	106214		ASRB	(R4)	; (R3)=000400,CC=0011
3113	010602	103002		BCC	ASRB1A	
3114	010604	102001		BVC	ASRB1A	
3115	010606	001001		BNE	.+4	



# MO6

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58 T6

MACY11 27(1006) 07-FEB-77 10:08 PAGE 77  
 CHECK UNIARY BYTE INSTRUCTIONS USING ADDRESS MODE 1

3116	010610	104000	ASRB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3117					
3118	010612	105314	DEC8	(R4)	;(R3)=000000,CC=0100
3119	010614	001401	BEQ	+.4	
3120	010616	104000	HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3121					
3122	010620	000261	SEC		
3123	010622	106014	RORB	(R4)	;(R3)=100000,CC=1010
3124	010624	103402	BCS	RORB1A	
3125	010626	102001	BVC	RORB1A	
3126	010630	100401	BMI	+.4	
3127	010632	104000	RORB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3128					
3129	010634	000242	CLV		
3130	010636	105314	DEC8	(R4)	;(R3)=077400,CC=0100
3131	010640	102401	BVS	+.4	
3132	010642	104000	HLT		
3133					
3134	010644	000261	SEC		
3135	010646	105313	DEC8	(R3)	;(R3)=077777,CC=1001
3136	010650	103002	BCC	DEC81A	
3137	010652	102401	BVS	DEC81A	
3138	010654	100401	BMI	+.4	
3139	010656	104000	DEC81A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3140					
3141	010660	000277	SCC		
3142	010662	000313	SWAB	(R3)	;(R3)=177577=[1774][177],CC=0000
3143	010664	103402	BCS	SWAB1	
3144	010666	102401	BVS	SWAB1	
3145	010670	100001	BPL	+.4	
3146	010672	104000	SWAB1: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3147					
3148	010674	105714	TSTB	(R4)	;(R3)=177577=[1774][177],CC=1000
3149	010676	103402	BCS	TSTB1	
3150	010700	102401	BVS	TSTB1	
3151	010702	100401	BMI	+.4	
3152	010704	104000	TSTB1: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3153					
3154	010706	105014	CLRB	(R4)	;(R3)=000177=[0000][177],CC=0100
3155	010710	001401	BEQ	+.4	
3156	010712	104000	HLT		
3157	010714	106313	ASLB	(R3)	;(R3)=000376,CC=1010
3158	010716	103402	BCS	ASLB1A	
3159	010720	102001	BVC	ASLB1A	
3160	010722	100401	BMI	+.4	
3161	010724	104000	ASLB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3162					
3163	010726	105113	COMB	(R3)	;(R3)=000001,CC=0001
3164	010730	103002	BCC	COMB1A	
3165	010732	102401	BVS	COMB1A	
3166	010734	100001	BPL	+.4	
3167	010736	104000	COMB1A: HLT		;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3168					
3169	010740	000313	SWAB	(R3)	;(R3)=000400,CC=0100
3170	010742	001401	BEQ	+.4	
3171	010744	104000	HLT		



```

3172
3173 010746 105213          INCB      (R3)
3174 010750 000261          SEC
3175 010752 105613          SBCB      (R3)          ;(R3)=000400,CC=0100
3176 010754 001401          BEQ      .+4
3177 010756 104000          HLT
3178 010760 022713 000400    CMP      #400,(R3)      ;CHECK REMAINING RESULT
3179 010764 001401          BEQ      .+4
3180 010766 104000          HLT
3181
3182
3183
3184 010770
3185 010770 000004          ;*****
3186 010772 112737 000007 001202    ;#TEST 7 CHECK UNIARY WORD OPS USING ADDRESS MODES 2 & 4
3187 011000 000401          ;*****
3188 011002 000000          TST7:
3189 011004 010704          SCOPE
3190 011006 162704 000004    MOV      #7,2#STSTNM
3191 011012 010405          BR       .+4
3192 011014 005015          .WORD   0          ;ADDRESS RESERVED FOR TESTS
3193
3194 011016 000277          MOV      PC,R4
3195 011020 000244          SUB      #4,R4      ;R4 AND R5 POINT TO
3196 011022 005725          MOV      R4,R5      ;RESERVED WORD
3197 011024 103402          BCS     TEST2
3198 011026 102401          BVS     TEST2
3199 011030 001401          BEQ     .+4
3200 011032 104000          TEST2: HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3201
3202 011034 005145          COM     -(R5)
3203 011036 103001          BCC     COM4
3204 011040 100401          BMI     .+4
3205 011042 104000          COM4:  HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3206
3207 011044 000241          CLC
3208 011046 006024          ROR     (R4)+
3209 011050 103002          BCC     ROR2
3210 011052 102001          BVC     ROR2
3211 011054 100001          BPL     .+4
3212 011056 104000          ROR2:  HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3213
3214 011060 000257          CCC
3215 011062 005244          INC     -(R4)
3216 011064 102002          BVC     INC4
3217 011066 001401          BEQ     INC4
3218 011070 100401          BMI     .+4
3219 011072 104000          INC4:  HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3220
3221 011074 000261          SEC
3222 011076 000324          SWAB   (R4)+
3223 011103 103401          BCS     SWAB2
3224 011102 100401          BMI     .+4
3225 011104 104000          SWAB2: HLT          ;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3226
3227 011106 005425          NEG     (R5)+

```

3228	011110	103001	BCC	NEG2	
3229	011112	100401	BMI	+.4	
3230	011114	104000	NEG2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3231					
3232	011116	005044	CLR	-(R4)	; (R4)=000000,CC=0100
3233	011120	001401	BEQ	+.4	
3234	011122	104000	HLT		
3235					
3236	011124	000261	SEC		
3237	011126	006045	ROR	-(R5)	; (R5)=100000,CC=1010
3238	011130	000261	SEC		
3239	011132	005525	ADC	(R5)+	; (R5)=100001,CC=1000
3240	011134	102401	BVS	ADC2	
3241	011136	100401	BMI	+.4	
3242	011140	104000	ADC2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3243					
3244	011142	000262	SEV		
3245	011144	006224	ASR	(R4)+	; (R4)=140000,CC=1001
3246	011146	103002	BCC	ASR2	
3247	011150	102401	BVS	ASR2	
3248	011152	100401	BMI	+.4	
3249	011154	104000	ASR2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3250					
3251	011156	000262	SEV		
3252	011160	006144	ROL	-(R4)	; (R4)=100001, CC=1001
3253	011162	103002	BCC	ROL4	
3254	011164	102401	BVS	ROL4	
3255	011166	100401	BMI	+.4	
3256	011170	104000	ROL4:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3257					
3258	011172	005645	SBC	-(R5)	; (R5)=100000,CC=1000
3259	011174	103001	BCC	+.4	
3260	011176	104000	HLT		;ERROR! 'C' BIT FAILED TO CLEAR
3261					
3262	011200	005325	DEC	(R5)+	; (R5)=077777,CC=0010
3263	011202	103402	BCS	DEC2	
3264	011204	102001	BVC	DEC2	
3265	011206	100001	BPL	+.4	
3266	011210	104000	DEC2:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3267					
3268	011212	006324	ASL	(R4)+	; (R4)=177776,CC=1010
3269	011214	102401	BVS	+.4	
3270	011216	104000	HLT		
3271	011220	006344	ASL	-(R4)	; (R4)=177774,CC=1001
3272	011222	103003	BCC	ASL4	
3273	011224	102402	BVS	ASL4	
3274	011226	001401	BEQ	ASL4	
3275	011230	100401	BMI	+.4	
3276	011232	104000	ASL4:	HLT	;ERROR! INCORRECT CC'S AS SHOWN ABOVE
3277					
3278	011234	022724	CMP	#177774,(R4)+	
3279	011240	001401	BEQ	+.4	
3280	011242	104000	HLT		
3281	011244	020405	CMP	R4,R5	
3282	011246	001401	BEQ	+.4	
3283	011250	104000	HLT		



```

3284
3285
3286
3287 011252
3288 011252 000004
3289 011254 112737 000010 001202
3290 011262 000401
3291 011264 000000
3292 011266 010705
3293 011270 162705 000004
3294 011274 010500
3295 011276 010002
3296 011300 005202
3297 011302 005010
3298
3299 011304 000277
3300 011306 000241
3301 011310 105125
3302 011312 103002
3303 011314 102401
3304 011316 100401
3305 011320 104000 COMB2:
3306
3307 011322 105542 ADCB -(R2)
3308 011324 001401 BEQ .+4
3309 011326 104000 HLT
3310 011330 105525 ADCB (R5)+
3311 011332 103401 BCS ADCB2
3312 011334 001001 BNE .+4
3313 011336 104000 ADCB2: HLT
3314
3315 011340 000263 +SEC!SEV
3316 011342 106045 RORB -(R5)
3317 011344 103003 BCC RORB4
3318 011346 102402 BVS RORB4
3319 011350 001401 BEQ RORB4
3320 011352 100401 BMI .+4
3321 011354 104000 RORB4: HLT
3322
3323 011356 000277 SCC
3324 011360 106122 ROLB (R2)+
3325 011362 103403 BCS ROLB2
3326 011364 102402 BVS ROLB2
3327 011366 001401 BEQ ROLB2
3328 011370 100001 BPL .+4
3329 011372 104000 ROLB2: HLT
3330
3331 011374 000257 CCC
3332 011376 106225 ASRB (R5)+
3333 011400 103402 BCS ASRB2
3334 011402 102001 BVC ASRB2
3335 011404 100401 BMI .+4
3336 011406 104000 ASRB2: HLT
3337
3338 011410 105242 INCB -(R2)
3339 011412 000277 SCC
    
```

```

*****
;TEST 10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4
*****
TST10:
    
```

```

;RESERVE A WORD
;RESERVED WORD
;R5 POINTS TO EVEN BYTE OF RESERVED WORD
;R2 POINTS TO ODD BYTE OF RESERVED WORD
;PRESET
; (R0)=000377,CC=1001
;ERROR! INCORRECT CC'S AS SHOWN ABOVE
; (R0)=000000,CC=0101
;ERROR! INCORRECT RESULT AS SHOWN ABOVE
; (R0)=000400,CC=0000
;ERROR! INCORRECT CC'S AS SHOWN ABOVE
; (R0)=100000,CC=1001
;ERROR! INCORRECT CC'S AS SHOWN ABOVE
; (R0)=100001,CC=0000
;ERROR! INCORRECT CC'S AS SHOWN ABOVE
; (R0)=140001, CC=1010
;ERROR! INCORRECT CC'S AS SHOWN ABOVE
; (R0)=140002,CC=0000
    
```



3340	011414	106222	ASRB	(R2)+	; (R0)=140001,CC=0000
3341	011416	103402	BCS	ASRB2A	
3342	011420	102401	BVS	ASRB2A	
3343	011422	100001	BPL	.+4	
3344	011424	104000	ASRB2A:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
3345					
3346	011426	000266		+SEZ!SEV	; SET Z,V
3347	011430	106345	ASLB	-(R5)	; (R0)=100001,CC=1001
3348	011432	103003	BCC	ASLB4	
3349	011434	102402	BVS	ASLB4	
3350	011436	001401	BEQ	ASLB4	
3351	011440	100401	BMI	.+4	
3352	011442	104000	ASLB4:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
3353					
3354	011444	105322	DECB	(R2)+	; (R0)=077401=[0774][001] ,CC=0010
3355	011446	103002	BCC	DECB2	
3356	011450	102001	BVC	DECB2	
3357	011452	100001	BPL	.+4	
3358	011454	104000	DECB2:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
3359					
3360	011456	105645	SBCB	-(R5)	; (R0)=077400, CC=0100
3361	011460	103402	BCS	SBCB4	
3362	011462	102401	BVS	SBCB4	
3363	011464	001401	BEQ	.+4	
3364	011466	104000	SBCB4:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
3365					
3366	011470	105442	NEGB	-(R2)	; (R0)=10400,CC=1001
3367	011472	103002	BCC	NEGB4	
3368	011474	102401	BVS	NEGB4	
3369	011476	100401	BMI	.+4	
3370	011500	104000	NEGB4:	HLT	; ERROR! INCORRECT CC'S AS SHOWN ABOVE
3371					
3372	011502	105725	TSTB	(R5)+	; (R0)=100400,CC=0100
3373	011504	103401	BCS	TSTB2	
3374	011506	001401	BEQ	.+4	
3375	011510	104000	TSTB2:	HLT	
3376					
3377	011512	105722	TSTB	(R2)+	; (R0)=100400,CC=1000
3378	011514	001401	BEQ	TSTB2A	
3379	011516	100401	BMI	.+4	
3380	011520	104000	TSTB2A:	HLT	
3381					
3382	011522	000261	SEC		
3383	011524	000342	SWAB	-(R2)	; (R0)=000201,CC=1000
3384	011526	103401	BCS	SWAB4	
3385	011530	100401	BMI	.+4	
3386	011532	104000	SWAB4:	HLT	
3387					
3388	011534	000277	SCC		
3389	011536	105225	INCB	(R5)+	; (R0)=000601=[0004][201],CC=0000
3390	011540	103003	BCC	INCB2	
3391	011542	102402	BVS	INCB2	
3392	011544	001401	BEQ	INCB2	
3393	011546	100001	BPL	.+4	
3394	011550	104000	INCB2:	HLT	
3395					

E07

MAINDEC-11-DGKDC-A PDP 11/6X SERIES CPU EXERCISER  
DGKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 82  
T10 CHECK UNIARY BYTE OPS USING ADDRESS MODES 2 & 4

3396	011552	022227	000601	CMP	(R2)+, #000601	;CHECK END RESULT
3397	011556	001401		BEQ	.+4	
3398	011560	104000		HLT		
3399	011562	020205		CMP	R2, R5	;CHECK REGISTERS
3400	011564	001401		BEQ	.+4	
3401	011566	104000		HLT		

```

*****
;TEST 11 CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5
*****
TST11:

```

3405	011570			SCOPE		
3406	011570	000004		MOV	#11, @#STSTNM	
3407	011572	112737	000011 001202	BR	.+6	;RESERVE 2 WORDS
3408	011600	000402		.WORD	0	;1 FOR THE ADDRESS
3409	011602	000000		.WORD	0	;AND 1 FOR DATA
3410	011604	000000		MOV	PC, R3	
3411	011606	010703		SUB	#4, R3	
3412	011610	162703	000004	CLR	(R3)	;PRESET DATA
3413	011614	005013		MOV	R3, R0	;R0 POINTS TO DATA WORD
3414	011616	010300		TST	-(R3)	
3415	011620	005743		MOV	R0, (R3)	
3416	011622	010013		MOV	R3, R4	
3417	011624	010304				
3418						
3419	011626	000257		CCC		
3420	011630	005733		TST	@(R3)+	; (R0)=000000, CC=0100
3421	011632	001401		BEQ	.+4	
3422	011634	104000		HLT		
3423						
3424	011636	000261		SEC		
3425	011640	006053		ROR	@-(R3)	; (R0)=100000, CC=1010
3426	011642	103402		BCS	ROR5	
3427	011644	102001		BVC	ROR5	
3428	011646	100401		BMI	.+4	
3429	011650	104000		HLT		
3430						
3431	011652	000257		CCC		
3432	011654	006234		ASR	@(R4)+	; (R0)=140000, CC=1010
3433	011656	102001		BVC	ASR3	
3434	011660	100401		BMI	.+4	
3435	011662	104000		HLT		
3436						
3437	011664	000250		CLN		
3438	011666	006333		ASL	@(R3)+	; (R0)=100000, CC=1001
3439	011670	103002		BCC	ASL3	
3440	011672	102401		BVS	ASL3	
3441	011674	100401		BMI	.+4	
3442	011676	104000		HLT		
3443						
3444	011700	000277		SCC		
3445	011702	005354		DEC	@-(R4)	; (R0)=077777, CC=0010
3446	011704	103003		BCC	DEC5	
3447	011706	102002		BVC	DEC5	
3448	011710	001401		BEQ	DEC5	
3449	011712	100001		BPL	.+4	
3450	011714	104000		HLT		
3451						

# F07

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 83  
CHECK UNIARY WORD OPS USING ADDRESS MODES 3 & 5

```

3452 011716 005453      NEG      2-(R3)          ;(R0)=100001, CC=1001
3453 011720 103002      BCC      NEG5
3454 011722 102401      BVS      NEG5
3455 011724 100401      BMI      .+4
3456 011726 104000      HLT
NEG5:
3458 011730 000262      SEV
3459 011732 005134      COM      2(R4)+          ;(R0)=077776, CC=0001
3460 011734 103001      BCC      COM3
3461 011736 102001      BVC      .+4
3462 011740 104000      HLT
COM3:
3464 011742 005233      INC      2(R3)+          ;(R0)=077777, CC=0001
3465 011744 103001      BCC      INC3
3466 011746 100001      BPL      .+4
3467 011750 104000      HLT
INC3:
3469 011752 005554      ADC      2-(R4)          ;(R0)=100000, CC=1010
3470 011754 103402      BCS      ADC5
3471 011756 102001      BVC      ADC5
3472 011760 100401      BMI      .+4
3473 011762 104000      HLT
ADC5:
3475 011764 000257      CCC
3476 011766 006134      ROL      2(R4)+          ;(R0)=000000, CC=0111
3477 011770 103002      BCC      ROL3
3478 011772 102001      BVC      ROL3
3479 011774 001401      BEQ      .+4
3480 011776 104000      HLT
ROL3:
3482 012000 005253      INC      2-(R3)          ;(R0)=000001, CC=0001
3483 012002 005654      SBC      2-(R4)          ;(R0)=000000, CC=0100
3484 012004 103401      BCS      SBC5
3485 012006 001401      BEQ      .+4
3486 012010 104000      HLT
SBC5:
;*****
;TEST 12      CHECK UNIARY BYTE OPS USING ADDRESS MODES 3 & 5
;*****
TST12:
3491 012012 000004      SCOPE
3492 012014 112737      MOV      #12,2#STSTNM
3493 012022 000403      BR      .+10
3494 012024 000000      .WORD   0
3495 012026 000000      .WORD   0
3496 012030 000000      .WORD   0
3497 012032 010702      MOV      PC,R2
3498 012034 005742      TST     -(R2)
3499 012036 005742      TST     -(R2)
3500 012040 010200      MOV      R2,R0
3501 012042 005010      CLR     (R0)
3502 012044 005742      TST     -(R2)
3503 012046 005742      TST     -(R2)
3504 012050 010022      MOV      R0,(R2)+
3505 012052 005200      INC     R0
3506 012054 010022      MOV      R0,(R2)+
3507 012056 010200      MOV      R2,R0
;RESERVE 3 WORDS
;1 FOR EVEN BYTE ADDRESS
;1 FOR ODD BYTE ADDRESS
;AND 1 FOR DATA
;BACK R2 UP TO
;DATA WORD
;R0 POINTS TO THE DATA WORD
;PRESET DATA
;BACK R2 UP TO
;EVEN BYTE ADDRESS WORD
;LOAD ADDRESS
;ODD BYTE ADDRESS
;LOAD ODD BYTE ADDRESS
;RESET R0

```



3508	012060	010205	MOV	R2, R5	
3509	012062	105152	COMB	2-(R2)	; (R0)=177400, CC=1001
3510	012064	103001	BCC	COMB5	
3511	012066	100401	BMI	.+4	
3512	012070	104000	COMB5:	HLT	
3513	012072	105752	TSTB	2-(R2)	; (R0)=177400, CC=0100
3514	012074	001401	BEQ	.+4	
3515	012076	104000	HLT		
3516	012100	000262	SEV		
3517	012102	106255	ASRB	2-(R5)	; (R0)=177400, CC=1001
3518	012104	103002	BCC	ASRB5	
3519	012106	102401	BVS	ASRB5	
3520	012110	100401	BMI	.+4	
3521	012112	104000	ASRB5:	HLT	
3522					
3523	012114	105232	INCB	2(R2)+	; (R0)=177401, CC=000
3524	012116	103001	BCC	INCB3	
3525	012120	100001	BPL	.+4	
3526	012122	104000	INCB3:	HLT	
3527					
3528	012124	000241	CLC		
3529	012126	106055	RORB	2-(R5)	; (R0)=177400, CC=0111
3530	012130	103003	BCC	RORB5	
3531	012132	102002	BVC	RORB5	
3532	012134	001001	BNE	RORB5	
3533	012136	100001	BPL	.+4	
3534	012140	104000	RORB5:	HLT	
3535					
3536	012142	106332	ASLB	2(R2)+	; (R0)=177000, CC=1001
3537	012144	103002	BCC	ASLB3	
3538	012146	102401	BVS	ASLB3	
3539	012150	100401	BMI	.+4	
3540	012152	104000	ASLB3:	HLT	
3541					
3542	012154	105552	ADCB	2-(R2)	; (R0)=177400, CC=1000
3543	012156	103401	BCS	ADCB5	
3544	012160	100401	BMI	.+4	
3545	012162	104000	ADCB5:	HLT	
3546					
3547	012164	000277	SCC		
3548	012166	106135	ROLB	2(R5)+	; (R0)=177401, CC=0000
3549	012170	101402	BLOS	ROLB3	; BRANCH IF C OR Z IS SET
3550	012172	102401	BVS	ROLB3	
3551	012174	100001	BPL	.+4	
3552	012176	104000	ROLB3:	HLT	
3553					
3554	012200	000352	SWAB	2-(R2)	; (R0)=000777, CC=1000
3555	012202	100401	BMI	.+4	
3556	012204	104000	HLT		
3557					
3558	012206	000261	SEC		
3559	012210	105635	SBCB	2(R5)+	; (R0)=000377, CC=0100
3560	012212	103401	BCS	SBCB3	
3561	012214	001401	BEQ	.+4	
3562	012216	104000	SBCB3:	HLT	
3563					

3564 012220 105432  
3565 012222 105352  
3566 012224 103001  
3567 012226 001401  
3568 012230 104000

NEGB @ (R2)+ ; (RO)=000001  
DECB @-(R2) ; (RO)=000000, CC=0101  
BCC DECBS  
BEQ .+4  
DECBS: HLT

3569  
3570  
3571  
3572 012232  
3573 012232 000004  
3574 012234 112737 000013 001202  
3575 012242 005027  
3576 012244 000000  
3577 012246 010700  
3578 012250 024040  
3579 012252 000277  
3580 012254 006167 177764  
3581 012260 103403  
3582 012262 102402  
3583 012264 001401  
3584 012266 100001  
3585 012270 104000

\*\*\*\*\*  
: \*TEST 13 CHECK UNIARY WORD OPS USING ADDRESS MODE 6 (PC)  
: \*\*\*\*\*  
†ST13:

3586  
3587 012272 005167 177746  
3588 012276 103002  
3589 012300 102401  
3590 012302 100401  
3591 012304 104000  
3592 012306 006267 177732  
3593 012312 103402  
3594 012314 102001  
3595 012316 100401  
3596 012320 104000  
3597

SCOPE  
MOV @13, @#STSTNM  
CLR (PC)+ ; PRESET DATA = 0  
UWM6: .WORD 0 ; RESERVED FOR DATA  
MOV PC, RO  
CMP -(RO), -(RO) ; RO POINTS TO DATA WORD  
SCC  
ROL UWM6 ; (RO)=000001, CC=0000  
BCS ROL6  
BVS ROL6  
BEQ ROL6  
BPL .+4  
ROL6: HLT

3598 012322 000277  
3599 012324 005467 177714  
3600 012330 103003  
3601 012332 102402  
3602 012334 001401  
3603 012336 100001  
3604 012340 104000  
3605

COM UWM6 ; (RO)=177776, CC=1001  
BCC COM6  
BVS COM6  
BMI .+4  
COM6: HLT  
ASR UWM6 ; (RO)=177777, CC=1010  
BCS ASR6  
BVC ASR6  
BMI .+4  
ASR6: HLT

3606 012342 000277  
3607 012344 006067 177674  
3608 012350 103003  
3609 012352 102402  
3610 012354 001401  
3611 012356 100401  
3612 012360 104000  
3613

SCC  
NEG UWM6 ; (RO)=000001, CC=0001  
BCC NEG6  
BVS NEG6  
BEQ NEG6  
BPL .+4  
NEG6: HLT

3614 012362 005667 177656  
3615 012366 103402  
3616 012370 102001  
3617 012372 100001  
3618 012374 104000  
3619

ROR UWM6 ; (RO)=100000, CC=1001  
ROR ROR6  
BCC ROR6  
BVS ROR6  
BEQ ROR6  
BMI .+4  
ROR6: HLT

3614 012362 005667 177656  
3615 012366 103402  
3616 012370 102001  
3617 012372 100001  
3618 012374 104000  
3619

SBC UWM6 ; (RO)=077777, CC=0010  
BCS SBC6  
BVC SBC6  
BPL .+4  
SBC6: HLT

```

3620 012376 000242          CLV
3621 012400 005267 177640  INC      UWM6          ;(RO)=100000, CC=1011
3622 012404 103403          BCS      INC6
3623 012406 102002          BVC      INC6
3624 012410 001401          BEQ      INC6
3625 012412 100401          BMI      .+4
3626 012414 104000          INC6:  HLT
3628 012416 006267 177622  ASR      UWM6          ;(RO)=140000, CC=1010
3629 012422 000261          SEC
3630 012424 006367 177614  ASL      UWM6          ;(RO)=100000, CC=1001
3631 012430 103002          BCC      ASL6
3632 012432 102401          BVS      ASL6
3633 012434 100401          BMI      .+4
3634 012436 104000          ASL6:  HLT
3636 012440 005367 177600  DEC      UWM6          ;(RO)=077777, CC=0011
3637 012444 103002          BCC      DEC6
3638 012446 102001          BVC      DEC6
3639 012450 100001          BPL      .+4
3640 012452 104000          DEC6:  HLT
3642 012454 005567 177564  ADC      UWM6          ;(RO)=100000, CC=1010
3643 012460 103402          BCS      ADC6
3644 012462 102001          BVC      ADC6
3645 012464 100401          BMI      .+4
3646 012466 104000          ADC6:  HLT
3647 012470 000242          CLV
3648 012472 000367 177546  SWAB     UWM6
3649 012476 100401          BMI      .+4
3650 012500 104000          HLT
3651 012502 022710 000200  CMP      #200, (RO)
3652 012506 001401          BEQ      .+4
3653 012510 104000          HLT

```

```

*****
; TEST 14 CHECK UNIARY BYTE OPS (EVEN/ODD) USING ADDRESS MODE 6 (PC)
*****

```

```

3654
3655
3656
3657 012512
3658 012512 000004          SCOPE
3659 012514 112737 000014 001202  MOVB     #14, 2#STSTNM
3660 012522 012700 013064          MOV      #UBM6, RO
3661 012526 063700 001550          ADD      2#FACTOR, RO      ;RO POINTS TO ADDRESS OF DATA
3662 012532 005067 000326          CLR      UBM6              ;CLEAR DATA
3663 012536 000277          SCC
3664 012540 000244          CLZ
3665 012542 105767 000316          TSTB     UBM6
3666 012546 103403          BCS      TSTB6
3667 012550 102402          BVS      TSTB6
3668 012552 001001          BNE      TSTB6
3669 012554 100001          BPL      .+4
3670 012556 104000          TSTB6: HLT
3671
3672 012560 000257          CCC
3673 012562 105767 000277          TSTB     UBM6+1          ;TEST ODD BYTE
3674 012566 001401          BEQ      .+4
3675 012570 104000          HLT

```



3676							
3677	012572	105667	000266	SBCB	UBM6		;(RO)=000000, CC=0100
3678	012576	103402		BCS	SBCB6		
3679	012600	102401		BVS	SBCB6		
3680	012602	001401		BEQ	.+4		
3681	012604	104000		SBCB6:	HLT		
3682							
3683	012606	000261		1S:	SEC		
3684	012610	105267	000250	INCB	UBM6		;LOOP UNTIL (RO)=077600, CC=1011
3685	012614	100403		BMI	2S		
3686	012616	105567	000243	ADCB	UBM6+1		;INCB INST INCREMENTS EVEN BYTE
3687	012622	000771		BR	1S		;ADCB INCREMENTS ODD BYTE
3688	012624	103001		2S:	BCC	INCB6	
3689	012626	102401		BVS	.+4		
3690	012630	104000		INCB6:	HLT		
3691							
3692	012632	106367	000226	ASLB	UBM6		;(RO)=077400, CC=0111
3693	012636	103003		BCC	ASLB6		
3694	012640	102002		BVC	ASLB6		
3695	012642	001001		BNE	ASLB6		
3696	012644	100001		BPL	.+4		
3697	012646	104000		ASLB6:	HLT		
3698							
3699	012650	000242		CLV			
3700	012652	105567	000207	ADCB	UBM6+1		;(RO)=100000, CC=1010
3701	012656	103402		BCS	ADCB6		
3702	012660	102001		BVC	ADCB6		
3703	012662	100401		BMI	.+4		
3704	012664	104000		ADCB6:	HLT		
3705							
3706	012666	000261		SEC			
3707	012670	106067	000171	RORB	UBM6+1		;(RO)=140000, CC=1010
3708	012674	103402		BCS	RORB6		
3709	012676	102001		BVC	RORB6		
3710	012700	100401		BMI	.+4		
3711	012702	104000		RORB6:	HLT		
3712							
3713	012704	105167	000154	COMB	UBM6		;(RO)=140377 CC=1001
3714	012710	103002		BCC	COMB6		
3715	012712	102401		BVS	COMB6		
3716	012714	100401		BMI	.+4		
3717	012716	104000		COMB6:	HLT		
3718							
3719	012720	000262		SEV			
3720	012722	105467	000137	NEGB	UBM6+1		;(RO)=040377, CC=0001
3721	012726	103002		BCC	NEGB6		
3722	012730	102401		BVS	NEGB6		
3723	012732	100001		BPL	.+4		
3724	012734	104000		NEGB6:	HLT		
3725							
3726	012736	106167	000123	ROLB	UBM6+1		;(RO)=100777, CC=1010
3727	012742	103402		BCS	ROLB6		
3728	012744	102001		BVC	ROLB6		
3729	012746	100401		BMI	.+4		
3730	012750	104000		ROLB6:	HLT		
3731							

```

3732 012752 106267 000106      ASRB      UBM6      ;(RO)=100777, CC=1001
3733 012756 103002      BCC      ASRB6
3734 012760 102401      BVS      ASRB6
3735 012762 100401      BMI      .+4
3736 012764 104000      ASRB6:   HLT
3737
3738 012766 105267 000072      INCB      UBM6      ;(RO)=100400, CC=0101
3739 012772 103002      BCC      INCB6A
3740 012774 102401      BVS      INCB6A
3741 012776 001401      BEQ      .+4
3742 013000 104000      INCB6A:  HLT
3743
3744 013002 105367 000057      DECB      UBM6+1    ;(RO)=100000, CC=1001
3745 013006 103003      BCC      DECB6A
3746 013010 102402      BVS      DECB6A
3747 013012 001401      BEQ      DECB6A
3748 013014 100401      BMI      .+4
3749 013016 104000      DECB6A:  HLT
3750
3751 013020 000367 000040      SWAB      UBM6      ;(RO)=000200, CC=1000
3752 013024 103401      BCS      SWAB6
3753 013026 100401      BMI      .+4
3754 013030 104000      SWAB6:   HLT
3755
3756 013032 106167 000026      ROLB      UBM6      ;(RO)=000000, CC=0111
3757 013036 103002      BCC      ROLB6A
3758 013040 102001      BVC      ROLB6A
3759 013042 001401      BEQ      .+4
3760 013044 104000      ROLB6A:  HLT
3761
3762 013046 005767 000012      TST       UBM6      ;(RO)=000000, CC=0100
3763 013052 103402      BCS      TEST6
3764 013054 102401      BVS      TEST6
3765 013056 001401      BEQ      .+4
3766 013060 104000      TEST6:   HLT
3767
3768 013062 000401      BR        .+4      ;RESERVE A WORD
3769 013064 000000      UBM6:    .WORD      0      ;WORD RESERVED FOR DATA
3770 013066 000004      RELE1:   SCOPE
3771 013070 010702      MOV      PC,R2
3772 013072 062702 000012      ADD      #12,R2
3773 013076 012707 036574      MOV      #RELOC,PC ;GO RELOCATE PROGRAM CODE
3774 013102 000000      REL11:   .WORD      0
3775 ;1111111111111111 LAST ADDRESS OF CODE TO BE RELOCATED 1111111111
3776
3777 ;*****
3778 ;*TEST 15 CHECK UNIARY WORD OPS USING ADDRESS MODE 7
3779 ;*****
3780 013104      TST15:
3781 013104 012767 000001 166212      MOV      #1,STIMES ;;DO 1 ITERATION
3782 013112 000004      SCOPE
3783 013114 112737 000015 001202      MOVB     #15,#STSTNM
3784
3785 ;.SBTTL START OF SECTION 2
3786 ;2222222222222222 FIRST ADDRESS TO BE RELOCATED 22222222
3787 013122 112737 000015 001202      REL2:    MOVB     #STN-1,#STSTNM

```

3788	013130	010700		MOV	PC,RO	;GET PC
3789	013132	005740		TST	-(RO)	;RO CONTAINS THE ADDRESS OF REL2
3790	013134	010037	001554	MOV	RO,@#FRSTAD	;SAVE
3791	013140	010700		MOV	PC,RO	;GET CURRENT PC
3792	013142	162700	013142	SUB	#,RO	;SUBTRACT RELOCATION FACTOR
3793	013146	010037	001550	MOV	RO,@#FACTOR	;SAVE RELOCATION FACTOR
3794	013152	010737	001212	MOV	PC,@#SLPERR	;SET LOOP ADDRESS
3795	013156	062737	000026 001212	ADD	#26,@#SLPERR	;ADJUST
3796	013164	013737	001212 001210	MOV	@#SLPERR,@#SLPADR	
3797	013172	105737	001544	TSTB	@#NEXEC	;BR IF TEST CODE TO BE EXECUTED
3798	013176	001402		BEQ	+.6	
3799	013200	000167	004062	JMP	RELE2	
3800	013204	000403		BR	UM7	;RESERVE 3 WORDS FOR ADDRESSES & DATA
3801	013206	000000		.WORD	0	;CONTAINS ADDRESS OF UWM7
3802	013210	000000		.WORD	0	;CONTAINS DATA
3803	013212	000000		.WORD	0	;CONTAINS ADDRESS OF UWM7
3804						
3805	013214	010700		MOV	PC,RO	
3806	013216	005740		TST	-(RO)	
3807	013220	005740		TST	-(RO)	
3808	013222	005040		CLR	-(RO)	;CLEAR TEST DATA
3809	013224	010002		MOV	RO,R2	
3810	013226	010240		MOV	R2,-(RO)	;SET UP ADDRESS
3811	013230	005720		TST	(RO)+	;MOVE RO TO NEXT ADDRESS
3812	013232	005720		TST	(RO)+	
3813	013234	010210		MOV	R2,(RO)	;SET NEXT ADDRESS
3814	013236	010200		MOV	R2,RO	;SET RO POINTING TO DATA
3815	013240	000277		SCC		
3816	013242	000244		CLZ		
3817	013244	005772	000002	TST	@2(2)	; (RO)=000000, CC=0100
3818	013250	001401		BEQ	+.4	
3819	013252	104000		HLT		
3820						
3821	013254	000277		SCC		
3822	013256	005672	177776	SBC	@-2(2)	; (RO)=177777, CC=1001
3823	013262	103002		BCC	SBC7	
3824	013264	102401		BVS	SBC7	
3825	013266	100401		BMI	+.4	
3826	013270	104000		HLT		
3827						
3828	013272	000277		SCC		
3829	013274	000241		CLC		
3830	013276	006372	000002	ASL	@2(2)	; (RO)=177776, CC=1001
3831	013302	103002		BCC	ASL7	
3832	013304	102401		BVS	ASL7	
3833	013306	100401		BMI	+.4	
3834	013310	104000		HLT		
3835						
3836	013312	000257		CCC		
3837	013314	005372	000002	DEC	@2(2)	; (RO)=177775, CC=1000
3838	013320	103402		BCS	DEC7	
3839	013322	102401		BVS	DEC7	
3840	013324	100401		BMI	+.4	
3841	013326	104000		HLT		
3842						
3843	013330	000262		SEV		



3844	013332	006272	177776	ASR	2-2(2)	;(RO)=177776, CC=1001
3845	013336	103002		BCC	ASR7	
3846	013340	102401		BVS	ASR7	
3847	013342	100401		BMI	.+4	
3848	013344	104000		ASR7:	HLT	
3849						
3850	013346	000241		CLC		
3851	013350	000262		SEV		
3852	013352	006072	177776	ROR	2-2(2)	;(RO)=077777, CC=0000
3853	013356	101402		BLOS	ROR7	;BRANCH IF C OR Z IS SET
3854	013360	102401		BVS	ROR7	
3855	013362	100001		BPL	.+4	
3856	013364	104000		ROR7:	HLT	
3857						
3858	013366	000262		SEV		
3859	013370	005472	000002	NEG	2(2)	;(RO)=100001, CC=1001
3860	013374	103002		BCC	NEG7	
3861	013376	102401		BVS	NEG7	
3862	013400	100401		BMI	.+4	
3863	013402	104000		NEG7:	HLT	
3864						
3865	013404	000250		CLN		
3866	013406	000372	177776	SWAB	2-2(2)	;(RO)=000600, CC=1000
3867	013412	103401		BCS	SWAB7	
3868	013414	100401		BMI	.+4	
3869	013416	104000		SWAB7:	HLT	
3870						
3871	013420	000262		SEV		
3872	013422	005172	000002	COM	2(2)	;(RO)=177177, CC=1001
3873	013426	103002		BCC	COM7	
3874	013430	102401		BVS	COM7	
3875	013432	100401		BMI	.+4	
3876	013434	104000		COM7:	HLT	
3877						
3878	013436	000372	000002	SWAB	2(2)	;(RO)=077776, CC=1000
3879	013442	100401		BMI	.+4	
3880	013444	104000		HLT		
3881						
3882	013446	000277		SCC		
3883	013450	005572	177776	ADC	2-2(2)	;(RO)=077777, CC=0000
3884	013454	103402		BCS	ADC7	
3885	013456	102401		BVS	ADC7	
3886	013460	100001		BPL	.+4	
3887	013462	104000		ADC7:	HLT	
3888						
3889	013464	005272	000002	INC	2(2)	;(RO)=100000, CC=1010
3890	013470	102001		BVC	INC7	
3891	013472	100401		BMI	.+4	
3892	013474	104000		INC7:	HLT	
3893						
3894	013476	000257		CCC		
3895	013500	006172	177776	ROL	2-2(2)	;(RO)=000000, CC=0111
3896	013504	103002		BCC	ROL7	
3897	013506	102001		BVC	ROL7	
3898	013510	001401		BEQ	.+4	
3899	013512	104000		ROL7:	HLT	

N07

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 91  
T16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7

3900  
3901  
3902  
3903  
3904  
3905  
3906

013514  
013514 000004  
013516 112737 000016 001202  
013524 012700 013210

::\*\*\*\*\*  
:TEST 16 CHECK UNIARY BYTE OPS USING ADDRESS MODE 7  
:\*\*\*\*\*  
↑ST16:

SCOPE  
MOVB #16,2#STSTNM  
MOV #UWM7,RO

2

3907	013530	063700	001550	ADD	2#FACTOR,RO	
3908	013534	010002		MOV	RO,R2	
3909	013536	010067	177450	MOV	RO,UWM7+2	
3910	013542	005720		TST	(RO)+	
3911	013544	005210		INC	(RO)	;WORD FOLLOWING UWM7 CONTAINS ADDRESS
3912	013546	005740		TST	-(RO)	;OF ODD BYTE, RO POINTS TO DATA WORD
3913	013550	005010		CLR	(RO)	;PRESET DATA
3914	013552	010067	177430	MOV	RO,UWM7-2	
3915				;NOTE: 2(2) REFERENCES THE ODD BYTE, AND 2-2(2) REFERENCES THE EVEN BYTE.		
3916						
3917	013556	000263		+SEC!SEV		;SET C AND V
3918	013560	105672	000002	SBCB	2(2)	; (RO)=177400, CC=1001
3919	013564	103003		BCC	SBCB7	
3920	013566	102402		BVS	SBCB7	
3921	013570	001401		BEQ	SBCB7	
3922	013572	100401		BMI	.+4	
3923	013574	104000		SBCB7:	HLT	
3924						
3925	013576	000277		SCC		;SET CONDITION CODES
3926	013600	105572	177776	ADCB	2-2(2)	; (RO)=177401, CC=0000
3927	013604	103403		BCS	ADCB7	
3928	013606	102402		BVS	ADCB7	
3929	013610	001401		BEQ	ADCB7	
3930	013612	100001		BPL	.+4	
3931	013614	104000		ADCB7:	HLT	
3932						
3933	013616	105172	177776	COMB	2-2(2)	; (RO)=177776, CC=1001
3934	013622	103002		BCC	COMB7	
3935	013624	102401		BVS	COMB7	
3936	013626	100401		BMI	.+4	
3937	013630	104000		COMB7:	HLT	
3938						
3939	013632	000241		CLC		;CLEAR CARRY
3940	013634	106072	000002	RORB	2(2)	; (RO)=077776, CC=0011
3941	013640	103002		BCC	RORB7	
3942	013642	102001		BVC	RORB7	
3943	013644	100001		BPL	.+4	
3944	013646	104000		RORB7:	HLT	
3945						
3946	013650	105272	000002	INCB	2(2)	; (RO)=100376, CC=1011
3947	013654	103002		BCC	INCB7	
3948	013656	102001		BVC	INCB7	
3949	013660	100401		BMI	.+4	
3950	013662	104000		INCB7:	HLT	
3951						
3952	013664	105372	177776	DECB	2-2(2)	; (RO)=100375, CC=1001
3953	013670	103002		BCC	DECB7	
3954	013672	102401		BVS	DECB7	
3955	013674	100401		BMI	.+4	
3956	013676	104000		DECB7:	HLT	
3957						
3958	013700	106372	000002	ASLB	2(2)	; (RO)=000375, CC=0111
3959	013704	103002		BCC	ASLB7	
3960	013706	102001		BVC	ASLB7	
3961	013710	001401		BEQ	.+4	
3962	013712	104000		ASLB7:	HLT	



```

3963
3964 013714 000241          CLC          ;CLEAR CARRY
3965 013716 106272 177776  ASRB       @-2(2) ;(RO)=000376, CC=1001
3966 013722 103002          BCC       ASRB7
3967 013724 102401          BVS       ASRB7
3968 013726 100401          BMI       .+4
3969 013730 104000          ASRB7:    HLT
3970
3971 013732 105472 000002  NEG8      @2(2) ;(RO)=000376, CC=0100
3972 013736 103402          BCS      NEG87
3973 013740 102401          BVS      NEG87
3974 013742 001401          BEQ      .+4
3975 013744 104000          NEG87:   HLT
3976
3977 013746 000262          SEV
3978 013750 106172 177776  ROLB     @-2(2) ;(RO)=00374, CC=1001
3979 013754 103002          BCC     ROLB7
3980 013756 102401          BVS     ROLB7
3981 013760 100401          BMI     .+4
3982 013762 104000          ROLB7:  HLT
3983
3984 013764 105272 177776  INCB     @-2(2) ;(RO)=000375, CC=1001
3985 013770 105272 177776  INCB     @-2(2) ;(RO)=000376, CC=1001
3986 013774 105572 177776  ADCB     @-2(2) ;(RO)=000377, CC=1000
3987 014000 105172 177776  COMB     @-2(2) ;(RO)=000000, CC=0100
3988 014004 001401          BEQ     .+4
3989 014006 104000          HLT
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
014010
014010 000004
014012 112737 000017 001202
014020 000277
014022 010700
014024 103002
014026 102401
014030 001001
014032 104000
014034 010002
014036 000262
014040 160002
014042 103402
014044 102401
014046 001401
014050 104000
014052 000244
014054 010203
014056 103401
014060 001401
014062 104000
014064 000257
014066 000272

```

\*\*\*\*\*  
: \*TEST 17 CHECK BINARY OPS USING ADDRESS MODE 0  
\*\*\*\*\*  
†ST17:

```

SCOPE
MOV8   #17,@#STSTNM
SCC
MOV    PC,R0 ;SET CONDITION CODES
BCC   MOV0 ;RO=PC, CC=X001
BVS   MOV0
BNE   .+4
MOV0: HLT
MOV    R0,R2 ;R2=R0
SEV   ;SET V
SUB   R0,R2 ;R2=000000, CC=0100
BCS   SUB0
BVS   SUB0
BEQ   .+4
SUB0: HLT
CLZ
MOV    R2,R3 ;R2=R3=000000, CC=0100
BCS   MOV0A
BEQ   .+4
MOV0A: HLT
CCC
+SEV!SEN ;SET V & N

```

```

4019 014070 020203      CMP      R2,R3      ;R2=R3=000000, CC=0100
4020 014072 103403      BCS     CMPO
4021 014074 102402      BVS     CMPO
4022 014076 001001      BNE     CMPO
4023 014100 100001      BPL     .+4
4024 014102 104000      CMPO:  HLT
4025
4026 014104 010002      MOV     R0,R2      ;R0=R2
4027 014106 010203      MOV     R2,R3      ;R0=R2=R3
4028 014110 060203      ADD     R2,R3      ;R3=2*R0
4029 014112 006302      ASL     R2,R3      ;R2=2*R0
4030 014114 020203      CMP     R2,R3      ;R2=R3=2*R0
4031 014116 001401      BEQ     .+4
4032 014120 104000      HLT
4033
4034
4035
4036 014122 005002      ;THE FOLLOWING SUBTEST SHIFTS A BIT THROUGH R2 AND R5 AND DOES A
4037 014124 005202      ;BIT TEST (BIT) USING R2 AND R5.
4038 014126 000402      CLR     R2
4039 014130 006302      1$:    INC     R2
4040 014132 100407      BR      2$
4041 014134 010205      2$:    ASL     R2
4042 014136 000277      BMI     4$
4043 014140 030205      MOV     R2,R5
4044 014142 103002      SCC     R2,R5      ;R2=R5
4045 014144 102401      BIT     R2,R5
4046 014146 001370      BCC     3$
4047 014150 104000      BVS     3$
4048 014152 010205      BNE     1$
4049 014154 000257      3$:    HLT
4050 014156 030205      4$:    MOV     R2,R5
4051 014160 100401      CCC
4052 014162 104000      BIT     R2,R5
4053
4054 014164 005002      BMI     .+4
4055 014166 000277      HLT
4056 014170 050002      CLR     R2
4057 014172 103002      SCC
4058 014174 102401      BIS     R0,R2
4059 014176 001001      BCC     BISO
4060 014200 104000      BVS     BISO
4061
4062 014202 010003      BNE     .+4
4063 014204 000277      HLT
4064 014206 000244      BISO:  HLT
4065 014210 040003      MOV     R0,R3
4066 014212 103003      SCC
4067 014214 102402      CLZ
4068 014216 001001      BIC     R0,R3
4069 014220 100001      BCC     BICO
4070 014222 104000      BVS     BICO
4071
4072 014224 010004      BNE     BICO
4073 014226 005104      BPL     .+4
4074 014230 040004      HLT
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100

```

```

4075 014232 005104      COM      R4
4076 014234 020004      CMP      R0,R4
4077 014236 001401      BEQ      .+4
4078 014240 104000      HLT
4079
4080 014242 010004      MOV      R0,R4
4081 014244 005104      COM      R4
4082 014246 010403      MOV      R4,R3
4083 014250 050003      BIS      R0,R3
4084 014252 103001      BCC      BISOA
4085 014254 100401      BMI      .+4
4086 014256 104000      BISOA:  HLT
4087 014260 005203      INC      R3
4088 014262 001401      BEQ      .+4
4089 014264 104000      HLT
4090 014266 010304      MOV      R3,R4
4091 014270 005103      COM      R3
4092 014272 000261      SEC
4093 014274 006004      ROR      R4
4094 014276 060304      ADD      R3,R4
4095 014300 103003      BCC      ADD0
4096 014302 102002      BVC      ADD0
4097 014304 001401      BEQ      ADD0
4098 014306 100001      BPL      .+4
4099 014310 104000      ADD0:  HLT
4100 014312 010700      MOV      PC,R0
4101 014314 022020      CMP      (R0)+,(R0)+
4102 014316 020007      CMP      R0,PC
4103 014320 001401      BEQ      .+4
4104 014322 104000      HLT
4105
4106 014324 010700      MOV      PC,R0
4107 014326 062700 000010      ADD      #10,R0
4108 014332 010002      MOV      R0,R2
4109 014334 020700      CMP      PC,R0
4110 014336 001002      BNE      CMPOA
4111 014340 020200      CMP      R2,R0
4112 014342 001401      BEQ      .+4
4113 014344 104000      CMPOA:  HLT
4114
4115
4116
4117 014346
4118 014346 000004
4119 014350 112737 000020 001202      SCOPE
4120 014356 000402      MOV      #20,2#STSTNM
4121 014360 000000      BR      .+6
4122 014362 000000      .WORD 0
4123 014364 010704      .WORD 0
4124 014366 005744      MOV      PC,R4
4125 014370 005044      TST      -(R4)
4126 014372 010403      CLR      -(R4)
4127 014374 005043      MOV      R4,R3
4128
4129 014376 005113      CLR      -(R3)
4130 014400 005214      COM      (R3)
                          INC      (R4)

```

```

;R3=R4=0
;R3=177777
;SET C
;R4=100000
;R3=177777,R4=077777, CC=0011

```

```

;RESERVE TWO WORDS
;RESERVED FOR SOURCE DATA
;RESERVED FOR DESTINATION DATA

```

```

;R4 POINTS TO DESTINATION DATA
;R3 POINTS TO SOURCE DATA

```

```

;(R3)=177777
;(R4)=000001

```



# F08

4131	014402	000262	SEV		;SET V
4132	014404	061314	ADD	(R3),(R4)	; (R3)=177777,(R4)=000000, CC=0101
4133	014406	103002	BCC	ADD1	
4134	014410	102401	BVS	ADD1	
4135	014412	001401	BEQ	.+4	
4136	014414	104000	ADD1:	HLT	
4137					
4138	014416	000277	SCC		
4139	014420	000250	CLN		
4140	014422	021314	CMP	(R3),(R4)	; (R3)=177777,(R4)=000000, CC=1000
4141	014424	103403	BCS	CMP1	
4142	014426	102402	BVS	CMP1	
4143	014430	001401	BEQ	CMP1	
4144	014432	100401	BMI	.+4	
4145	014434	104000	CMP1:	HLT	
4146					
4147	014436	000277	SCC		
4148	014440	000244	CLZ		
4149	014442	031314	BIT	(R3),(R4)	; (R3)=177777,(R4)=000000, CC=0101
4150	014444	103002	BCC	BITT1	
4151	014446	102401	BVS	BITT1	
4152	014450	001401	BEQ	.+4	
4153	014452	104000	BITT1:	HLT	
4154					
4155	014454	000277	SCC		
4156	014456	000245	+CLC!CLZ		
4157	014460	005114	COM	(R4)	; (R4)=177777
4158	014462	161314	SUB	(R3),(R4)	; (R3)=177777,(R4)=000000, CC=0100
4159	014464	103402	BCS	SUB1	
4160	014466	102401	BVS	SUB1	
4161	014470	001401	BEQ	.+4	
4162	014472	104000	SUB1:	HLT	
4163					
4164	014474	105013	CLRB	(R3)	; (R3)=177400
4165	014476	000313	SWAB	(R3)	; (R3)=000377
4166	014500	000270	SEN		
4167	014502	011314	MOV	(R3),(R4)	; (R3)=(R4)=000377
4168	014504	100001	BPL	.+4	
4169	014506	104000	HLT		
4170	014510	000314	SWAB	(R4)	; (R3)=000377,(R4)=177400
4171	014512	000263	+SEC!SEV		;SET C & V
4172	014514	051314	BIS	(R3),(R4)	; (R3)=000377,(R4)=177777, CC=1001
4173	014516	103002	BCC	BIS1	
4174	014520	102401	BVS	BIS1	
4175	014522	100401	BMI	.+4	
4176	014524	104000	BIS1:	HLT	
4177					
4178	014526	041314	BIC	(R3),(R4)	; (R3)=000377,(R4)=177400, CC=1001
4179	014530	103002	BCC	BIC1	
4180	014532	102401	BVS	BIC1	
4181	014534	100401	BMI	.+4	
4182	014536	104000	BIC1:	HLT	
4183					
4184	014540	000262	SEV		;SET V
4185	014542	021314	CMP	(R3),(R4)	; (R3)=000377,(R4)=177400, CC=0001
4186	014544	103003	BCC	CMP1A	

```

4187 014546 102402          BVS    CMP1A
4188 014550 001401          BEQ    CMP1A
4189 014552 100001          BPL    .+4
4190 014554 104000          CMP1A: HLT
4191
4192 014556 005013          CLR    (R3)          ;(R3)=000000
4193 014560 000261          SEC
4194 014562 006013          ROR    (R3)          ;(R3)=100000
4195 014564 011314          MOV    (R3),(R4)    ;(R3)=(R4)=100000
4196 014566 005114          COM    (R4)          ;(R4)=077777
4197 014570 161314          SUB    (R3),(R4)    ;(R3)=100000,(R4)=177777, CC=1011
4198 014572 103002          BCC    SUB1A
4199 014574 102001          BVC    SUB1A
4200 014576 100401          BMI    .+4
4201 014600 104000          SUB1A: HLT
4202
4203 014602 000277          SCC
4204 014604 161314          SUB    (R3),(R4)    ;(R3)=100000,(R4)=077777, CC=0000
4205 014606 101402          BLOS  SUB1B        ;BRANCH IF C OR Z IS SET
4206 014610 102401          BVS    SUB1B
4207 014612 100001          BPL    .+4
4208 014614 104000          SUB1B: HLT
4209
4210 014616 011314          MOV    (R3),(R4)    ;(R3)=100000,(R4)=100000, CC=1000
4211 014620 001401          BEQ    MOV1
4212 014622 100401          BMI    .+4
4213 014624 104000          MOV1: HLT
4214
4215 014626 061314          ADD    (R3),(R4)    ;(R3)=100000,(R4)=000000, CC=0111
4216 014630 103003          BCC    ADD1A
4217 014632 102002          BVC    ADD1A
4218 014634 001001          BNE    ADD1A
4219 014636 100001          BPL    .+4
4220 014640 104000          ADD1A: HLT
4221
4222 014642 005113          COM    (R3)          ;(R3)=077777
4223 014644 011314          MOV    (R3),(R4)    ;(R4)=077777
4224 014646 061314          ADD    (R3),(R4)    ;(R3)=077777,(R4)=177776, CC=1010
4225 014650 103402          BCS    ADD1B
4226 014652 102001          BVC    ADD1B
4227 014654 100401          BMI    .+4
4228 014656 104000          ADD1B: HLT
4229
4230 014660 062714 000002          ADD    #2,(R4)
4231 014664 005714          TST   (R4)          ;CHECK FINAL RESULT
4232 014666 001401          BEQ    .+4
4233 014670 104000          HLT
4234
4235          ;*****
4236          ;*TEST 21 CHECK BINARY BYTE OPS USING ADDRESS MODE 1
4237          ;*****
4237 014672          TST21:
4238 014672 000004          SCOPE
4239 014674 112737 000021 001202          MOVB  #21,#STSTNM
4240 014702 000402          BR    .+6
4241 014704 000000          .WORD 0
4242 014706 000000          .WORD 0

```

# H08

MAJDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 98  
CHECK BINARY BYTE OPS USING ADDRESS MODE 1

```

4243 014710 010705      MOV      PC,R5
4244 014712 005745      TST      -(R5)
4245 014714 005045      CLR      -(R5)          ;(R5)=000000
4246 014716 010502      MOV      R5,R2
4247 014720 005042      CLR      -(R2)          ;(R2)=000000
4248 014722 005202      INC      R2              ;R2 POINTS TO ODD BYTE
4249 014724 105112      COMB     (R2)           ;(R2)=177400
4250
4251 014726 000277      SCC
4252 014730 111215      MOVB     (R2),(R5)     ;(R2)=177400,(R5)=000377,CC=1001
4253 014732 103005      BCC      MOVB1
4254 014734 102404      BVS      MOVB1
4255 014736 001403      BEQ      MOVB1
4256 014740 100002      BPL      MOVB1
4257 014742 105215      INCB     (R5)          ;CHECK RESULT
4258 014744 001401      BEQ      .+4
4259 014746 104000      MOVB1:  HLT
4260
4261 014750 106312      ASLB     (R2)          ;SHIFT (R2) UNTIL
4262 014752 102376      BVC      .-2           ;:(R2)=000000
4263 014754 106012      RORB     (R2)          ;:(R2)=100000
4264 014756 105315      DECB     (R5)          ;:(R5)=00377
4265 014760 106015      RORB     (R5)          ;:(R5)=000177
4266 014762 000257      CCC
4267 014764 121512      CMPB     (R5),(R2)     ;(R5)=000177,(R2)=100000,CC=1010
4268 014766 102001      BVC      CMPB1
4269 014770 100401      BMI      .+4
4270 014772 104000      CMPB1:  HLT
4271
4272 014774 005003      CLR      R3
4273 014776 000261      SEC
4274 015000 006003      ROR      R3            ;R3=100000
4275 015002 050315      BIS      R3,(R5)      ;:(R5)=100177
4276 015004 000273      +SEC!SEV!SEN         ;SET C V & N
4277 015006 131215      BITB     (R2),(R5)    ;:(R2)=100000,(R5)=100177,CC=0101
4278 015010 103002      BCC      BITB1
4279 015012 102401      BVS      BITB1
4280 015014 001401      BEQ      .+4
4281 015016 104000      BITB1:  HLT
4282
4283 015020 151215      BISB     (R2),(R5)     ;(R2)=100000,(R5)=100377,CC=1001
4284 015022 103001      BCC      BISB1
4285 015024 100401      BMI      .+4
4286 015026 104000      BISB1:  HLT
4287
4288 015030 141215      BICB     (R2),(R5)     ;(R2)=100000,(R5)=100177,CC=0001
4289 015032 103002      BCC      BICB1
4290 015034 001401      BEQ      BICB1
4291 015036 100001      BPL      .+4
4292 015040 104000      BICB1:  HLT
4293
4294 015042 105112      COMB     (R2)          ;(R2)=077400,(R5)=100177
4295 015044 121215      CMPB     (R2),(R5)
4296 015046 001401      BEQ      .+4
4297 015050 104000      HLT
4298

```



4299	015052	141512	BICB	(R5), (R2)	; (R5)=100177, (R2)=000000, CC=0100
4300	015054	001002	BNE	BICB1A	
4301	015056	105712	TSTB	(R2)	
4302	015060	001401	BEQ	+.4	
4303	015062	104000	BICB1A:	HLT	
4304					
4305	015064	000402	BR	+.6	; RESERVE TWO WORDS FOR DATA
4306	015066	000000	.WORD	0	; SOURCE DATA
4307	015070	000000	.WORD	0	; DEST DATA
4308	015072	010705	MOV	PC, R5	
4309	015074	005745	TST	-(R5)	
4310	015076	105045	CLRB	-(R5)	; R5 POINTS TO DEST ODD BYTE
4311	015100	010504	MOV	R5, R4	
4312	015102	105044	CLRB	-(R4)	; R4 POINTS TO DEST EVEN BYTE
4313	015104	010403	MOV	R4, R3	
4314	015106	105043	CLRB	-(R3)	; R3 POINTS TO SOURCE ODD BYTE
4315	015110	010302	MOV	R3, R2	
4316	015112	105042	CLRB	-(R2)	; R2 POINTS TO SOURCE EVEN BYTE
4317					
4318					
4319					
4320	015114	000261	SEC		; COMMENTS ARE LEAST SIGNIFICANT 4 BITS OF BYTES POINTED TO BY R2, R3 ; R4, AND R5 RESPECTIVELY AND THE REMAINING BITS ARE 0'S.
4321					; SET CARRY
4322	015116	106112	ROLB	(R2)	; (R2), (R3), (R4), (R5)
4323	015120	111214	MOVB	(R2), (R4)	; 0001, 0000, 0000, 0000
4324	015122	106112	ROLB	(R2)	; 0001, 0000, 0001, 0000
4325	015124	111213	MOVB	(R2), (R3)	; 0010, 0000, 0001, 0000
4326	015126	106112	ROLB	(R2)	; 0010, 0010, 0001, 0000
4327	015130	111315	MOVB	(R3), (R5)	; 0100, 0010, 0001, 0010
4328	015132	106112	ROLB	(R2)	; 0100, 0010, 0001, 0010
4329	015134	106113	ROLB	(R3)	; 1000, 0100, 0001, 0010
4330	015136	151215	BISB	(R2), (R5)	; 1000, 0100, 0001, 1010
4331	015140	131512	BITB	(R5), (R2)	; 1000, 0100, 0001, 1010
4332	015142	001426	BEQ	BIN1	
4333	015144	151314	BISB	(R3), (R4)	; 1000, 0100, 0101, 1010
4334	015146	131413	BITB	(R4), (R3)	; 1000, 0100, 0101, 1010
4335	015150	001423	BEQ	BIN1	
4336	015152	105213	INCB	(R3)	; 1000, 0101, 0101, 1010
4337	015154	121314	CMPB	(R3), (R4)	; 1000, 0101, 0101, 1010
4338	015156	001020	BNE	BIN1	
4339	015160	106113	ROLB	(R3)	; 1000, 1010, 0101, 1010
4340	015162	121315	CMPB	(R3), (R5)	; 1000, 1010, 0101, 1010
4341	015164	001015	BNE	BIN1	
4342	015166	106212	ASRB	(R2)	; 0100, 1010, 0101, 1010
4343	015170	131214	BITB	(R2), (R4)	; 0100, 1010, 0101, 1010
4344	015172	001412	BEQ	BIN1	
4345	015174	106015	RORB	(R5)	; 0100, 1010, 0101, 0101
4346	015176	121415	CMPB	(R4), (R5)	; 0100, 1010, 0101, 0101
4347	015200	001007	BNE	BIN1	
4348	015202	105314	DECB	(R4)	; 0100, 1010, 0100, 0101
4349	015204	141214	BICB	(R2), (R4)	; 0100, 1010, 0000, 0101
4350	015206	001004	BNE	BIN1	
4351	015210	111314	MOVB	(R3), (R4)	; 0100, 1010, 1010, 0101
4352	015212	106213	ASRB	(R3)	; 0100, 0101, 1010, 0101
4353	015214	141315	BICB	(R3), (R5)	; 0100, 0101, 1010, 0101
4354	015216	001401	BEQ	+.4	

# JOB

MAINDEC-11-DGKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DGKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 100  
 T21 CHECK BINARY BYTE OPS USING ADDRESS MODE 1

```

4355 015220 104000
4356
4357
4358
4359 015222
4360 015222 000004
4361 015224 112737 000022 001202
4362 015232 012704 015070
4363 015236 012702 015066
4364 015242 063702 001550
4365 015246 063704 001550
4366 015252 010405 ;SET DESTINATION REGISTER
4367 015254 012715 000001
4368 015260 012712 177777
4369 015264 000257
4370 015266 000262
4371 015270 062225 ;(R2)=177777,(R5)=000000,CC=0101
4372 015272 103002
4373 015274 102401
4374 015276 001401
4375 015300 104000 ADD2:
4376
4377 015302 000262
4378 015304 024527 000001 SEV ;SET V
4379 015310 103002 CMP -(R5),#1 ;(R5)=000000,CC=1001
4380 015312 102401 BCC
4381 015314 100401 BVS
4382 015316 104000 CMP2: BMI
4383
4384 015320 054225 BIS -(R2),(R5)+ ;(R2)=177777,(R5)=177777,CC=1001
4385 015322 103001 BCC BIS2
4386 015324 100401 BMI
4387 015326 104000 BIS2: HLT
4388 015330 000277 SCC
4389 015332 000244 CLZ
4390 015334 162245 SUB (R2)+,-(R5) ;(R2)=177777,(R5)=000000,CC=0100
4391 015336 103402 BCS
4392 015340 102401 BVS
4393 015342 001401 BEQ
4394 015344 104000 SUB2: HLT
4395
4396 015346 005442 NEG -(R2) ;(R2)=000001
4397 015350 005115 COM (R5) ;(R5)=177777
4398 015352 000277 SCC
4399 015354 000250 CLN
4400 015356 042225 BIC (R2)+,(R5)+ ;(R2)=000001,(R5)=177776,CC=1001
4401 015360 103003 BCC
4402 015362 102402 BVS
4403 015364 001401 BEQ
4404 015366 100401 BMI
4405 015370 104000 BIC2: HLT
4406
4407 015372 012742 125252 MOV #125252,-(R2)
4408 015376 012245 MOV (R2)+,-(R5)
4409 015400 005125 COM (R5)+ ;(R5)=052525
4410 015402 000262 SEV
  
```



K08

```

4411 015404 034245          BIT      -(R2),-(R5)      ;(R2)=125252,(R5)=052525, CC=0101
4412 015406 103002          BCC     BITT2
4413 015410 102401          BVS     BITT2
4414 015412 001401          BEQ     .+4
4415 015414 104000          BITT2: HLT
4416
4417 015416 000262          SEV
4418 015420 052225          BIS     (R2)+,(R5)+      ;(R2)=125252,(R5)=177777, CC=1001
4419 015422 103002          BCC     BIS2A
4420 015424 102401          BVS     BIS2A
4421 015426 100401          BMI     .+4
4422 015430 104000          BIS2A: HLT
4423
4424 015432 042745 125252          BIC     #125252,-(R5)    ;(R5)=052525
4425 015436 005125          COM     (R5)+            ;(R5)=125252
4426 015440 024245          CMP     -(R2),-(R5)
4427 015442 001401          BEQ     .+4
4428 015444 104000          HLT
4429
4430 015446 005012          CLR     (R2)
4431 015450 005122          COM     (R2)+            ;(R2)=177777
4432 015452 162742 000001          SUB     #1,-(R2)         ;(R2)=177776, CC=1000
4433 015456 103402          BCS     SUB2A
4434 015460 102401          BVS     SUB2A
4435 015462 100401          BMI     .+4
4436 015464 104000          SUB2A: HLT
4437 015466 010702          MOV     PC,R2            ;GET CURRENT PC
4438 015470 010205          MOV     R2,R5           ;MOVE TO R5
4439 015472 124245          IS:    CMPB    -(R2),-(R5) ;COMPARE ALL PREVIOUS MEMORY ADDRESSES
4440 015474 001401          BEQ     .+4
4441 015476 104000          HLT
4442 015500 020237 001554          CMP     R2,#FRSTAD      ;ERROR!
4443 015504 001372          BNE     IS              ;CHECK FOR LOW LIMIT
4444
4445          ;*****
4446          ;*TEST 23 CHECK BINARY BYTE OPS USING ADDRESS MODE 2 & 4
4447          ;*****
4448          †ST23:
4449          SCOPE
4450          MOVB    #23,#STSTNM
4451          BR     .+6        ;RESERVE TWO WORDS
4452          .WORD 0         ;SOURCE DATA
4453          .WORD 0         ;DESTINATION DATA
4454          MOV     PC,R3
4455          TST    -(R3)
4456
4457          ;FIRST CHECK AUTO INCREMENT/DECREMENT
4458          MOV     R3,R0
4459          MOV     R0,R2
4460          DEC     R2
4461          MOV     SP,R4
4462          MOV     SP,R5
4463          TST    -(R5)
4464          MOVB    -(R0),-(SP)
4465          CMP     R5,SP
4466          BNE     BINB
    
```



```

4467 015552 020200      CMP      R2,R0
4468 015554 001017      BNE     BINB
4469 015556 122026      CMPB   (R0)+,(SP)+
4470 015560 020406      CMP      R4,SP
4471 015562 001014      BNE     BINB
4472 015564 020003      CMP      R0,R3
4473 015566 001012      BNE     BINB
4474 015570 154640      BISB   -(SP),-(R0)
4475 015572 020506      CMP      R5,SP
4476 015574 001007      BNE     BINB
4477 015576 020200      CMP      R2,R0
4478 015600 001005      BNE     BINB
4479 015602 142620      BICB   (SP)+,(R0)+
4480 015604 020406      CMP      R4,SP
4481 015606 001002      BNE     BINB
4482 015610 020003      CMP      R0,R3
4483 015612 001401      BEQ     .+4
4484 015614 104000      HLT
4485 015616 010003      MOV     R0,R3
4486 015620 112743 000200      MOVB   #200,-(R3)
4487 015624 112743 000377      MOVB   #377,-(R3)      ;(R3)=100377
4488 015630 010304      MOV     R3,R4
4489 015632 112744 000177      MOVB   #177,-(R4)
4490 015636 112744 000000      MOVB   #0,-(R4)      ;(R4)=077400
4491 015642 001401      BEQ     .+4
4492 015644 104000      HLT
4493
4494 015646 152324      BISB   (R3)+,(R4)+      ;(R3)=100377,(R4)=077777
4495 015650 100401      BMI     .+4
4496 015652 104000      HLT
4497
4498 015654 122324      CMPB   (R3)+,(R4)+
4499 015656 103402      BCS    CMPB2
4500 015660 102001      BVC    CMPB2
4501 015662 100001      BPL    .+4
4502 015664 104000      HLT
4503
4504 015666 000261      SEC
4505 015670 134344      BITB   -(R3),-(R4)
4506 015672 103002      BCC    BITB2
4507 015674 102401      BVS    BITB2
4508 015676 001401      BEQ     .+4
4509 015700 104000      HLT
4510
4511 015702 000244      CLZ
4512 015704 144344      BICB   -(R3),-(R4)      ;(R3)=100377,(R4)=077400
4513 015706 001401      BEQ     .+4
4514 015710 104000      HLT
4515
4516
4517
4518 015712
4519 015712 000004
4520 015714 112737 000024 001202
4521 015722 000404
4522 015724 000000

```

BINB:

CMPB2:

BITB2:

```

*****
:TEST 24 CHECK BINARY WORD OPS USING ADDRESS MODES 3 & 5
*****
↑ST24:

```

```

:RESERVE SPACE FOR DATA AND ADDRESSES
:CONTAINS ADDRESS OF SOURCE DATA

```

# M08

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58 T24

MACY11 27(1006) 07-FEB-77 10:08 PAGE 103  
 CHECK BINARY WORD OPS USING ADDRESS MODES 3 & 5

4523	015726	000000		.WORD	0	;CONTAINS ADDRESS OF DEST DATA
4524	015730	000000		.WORD	0	;CONTAINS SOURCE DATA
4525	015732	000000		.WORD	0	;CONTAINS DEST DATA
4526	015734	010701		MOV	PC,R1	
4527	015736	010100	25:	MOV	R1,R0	;SET SCOPE PTR
4528	015740	024040		CMP	-(R0),-(R0)	;ADJUST R0
4529	015742	010005		MOV	R0,R5	;R5 POINTS TO DEST DATA
4530	015744	024545		CMP	-(R5),-(R5)	;SUB 4 FROM R5
4531	015746	010015		MOV	R0,(R5)	;R5 POINTS TO ADDRESS OF DEST DATA
4532	015750	01050?		MOV	R5,R2	
4533	015752	010004		MOV	R0,R4	;R4 POINTS TO DEST DATA
4534	015754	005740		TST	-(R0)	
4535	015756	010003		MOV	R0,R3	;R3 POINTS TO SOURCE DATA
4536	015760	010042		MOV	R0,-(R2)	;R2 POINTS TO ADDRESS OF SOURCE DATA
4537	015762	005013		CLR	(R3)	;PRESET SOURCE DATA
4538	015764	005014		CLR	(R4)	;PRESET DEST DATA
4539						
4540	015766	000277		SCC		
4541	015770	000244		CLZ		
4542	015772	163235		SUB	a(R2)+,a(R5)+	; (R3)=000000, (R4)=000000, CC=0100
4543	015774	103402		BCS	SUB3	
4544	015776	102401		BVS	SUB3	
4545	016000	001401		BEQ	.+4	
4546	016002	104000		HLT		
4547						
4548	016004	052752	100000	BIS	#100000,a-(R2)	; (R3)=100000
4549	016010	062755	000001	ADD	#1,a-(R5)	; (R4)=000001
4550	016014	163235		SUB	a(R2)+,a(R5)+	; (R3)=100000, (R4)=100001, CC=1011
4551	016016	103002		BCC	SUB3A	
4552	016020	102001		BVC	SUB3A	
4553	016022	100401		BMI	.+4	
4554	016024	104000		HLT		
4555						
4556	016026	005414		NEG	(R4)	; (R4)=077777
4557	016030	035255		BIT	a-(R2),a-(R5)	; (R3)=100000, (R4)=077777
4558	016032	001401		BEQ	.+4	
4559	016034	104000		HLT		
4560	016036	023235		CMP	a(R2)+,a(R5)+	
4561	016040	102401		BVS	.+4	
4562	016042	104000		HLT		
4563	016044	005152		COM	a-(R2)	
4564	016046	000257		CCC		
4565	016050	063255		ADD	a(R2)+,a-(R5)	
4566	016052	102001		BVC	ADD3	
4567	016054	100401		BMI	.+4	
4568	016056	104000		HLT		
4569	016060	000261		SEC		
4570	016062	045235		BIC	a-(R2),a(R5)+	; (R3)=077777, (R4)=100000
4571	016064	103001		BCC	BIC3	
4572	016066	100401		BMI	.+4	
4573	016070	104000		HLT		
4574						
4575	016072	005155		COM	a-(R5)	; (R4)=077777
4576	016074	023235		CMP	a(R2)+,a(R5)+	; (R3)=077777, (R4)=077777
4577	016076	001401		BEQ	.+4	
4578	016100	104000		HLT		



\*\*\*\*\*  
: TEST 25 CHECK BINARY BYTE OPS USING ADDRESS MODES 3 & 5  
: \*\*\*\*\*  
†ST25:

4579									
4580									
4581									
4582	016102	000004							
4583	016102	112737	000025	001202	SCOPE				
4584	016104	000406			MOV	#25,2#STSTNM			
4585	016112	000000			BR	15			: RESERVE SPACE FOR ADDRESS AND DATA
4586	016114	000000			.WORD	0			: CONTAINS ADDRESS OF SOURCE DATA (EVEN BYTE)
4587	016116	000000			.WORD	0			: CONTAINS ADDRESS OF SOURCE DATA (ODD BYTE)
4588	016120	000000			.WORD	0			: CONTAINS ADDRESS OF DEST DATA (EVEN BYTE)
4589	016122	000000			.WORD	0			: CONTAINS ADDRESS OF DEST DATA (ODD BYTE)
4590	016124	000000			.WORD	0			: CONTAINS SOURCE DATA
4591	016126	000000			.WORD	0			: CONTAINS DEST DATA
4592									
4593	016130	010700			15: MOV	PC,R0			
4594	016132	024040			CMP	-(R0),-(R0)			: R0=ADDRESS OF DEST DATA
4595	016134	010003			MOV	R0,R3			: R3
4596	016136	010305			MOV	R3,R5			: R5
4597	016140	005743			TST	-(R3)			: SUB 2 FROM R3
4598	016142	010043			MOV	R0, -(R3)			: R3 POINTS TO ADDRESS OF DEST DATA
4599	016144	005213			INC	(R3)			: ODD BYTE
4600	016146	010043			MOV	R0, -(R3)			: EVEN BYTE
4601	016150	010304			MOV	R3,R4			
4602	016152	005740			TST	-(R0)			: R0=ADDRESS OF SOURCE DATA
4603	016154	010044			MOV	R0, -(R4)			: R4 POINTS TO ADDRESS OF SOURCE DATA
4604	016156	005214			INC	(R4)			: ODD BYTE
4605	016160	010044			MOV	R0, -(R4)			: EVEN BYTE
4606									
4607	016162	000261			SEC				: SET CARRY
4608	016164	012734	177001		MOV	#177001,2(R4)+			
4609	016170	112734	000200		MOV	#200,2(R4)+			: SOURCE DATA=100001
4610	016174	115433			MOV	2-(R4),2(R3)+			
4611	016176	115433			MOV	2-(R4),2(R3)+			: DEST DATA=000600
4612	016200	103401			BCS	.+4			
4613	016202	104000			HLT				: ERROR! MOV DOES AFFECT C BIT IN PSW
4614	016204	022715	000600		CMP	#600,(R5)			: CHECK DEST DATA
4615	016210	001401			BEQ	.+4			
4616	016212	104000			HLT				: ERROR! INCORRECT RESULT
4617	016214	024343			CMP	-(R3),-(R3)			: POINT R4 BACK TO EVEN BYTE
4618	016216	153433			BISB	2(R4)+,2(R3)+			
4619	016220	153433			BISB	2(R4)+,2(R3)+			: DEST DATA=100601
4620	016222	022715	100601		CMP	#100601,(R5)			: CHECK RESULT
4621	016226	001401			BEQ	.+4			
4622	016230	104000			HLT				: ERROR! INCORRECT DEST DATA AFTER BISB
4623	016232	145453			BICB	2-(R4),2-(R3)			
4624	016234	145453			BICB	2-(R4),2-(R3)			
4625	016236	133433			BITB	2(R4)+,2(R3)+			
4626	016240	001002			BNE	BITB3			
4627	016242	135433			BITB	2-(R4),2(R3)+			
4628	016244	001001			BNE	.+4			
4629	016246	104000			BITB3: HLT				
4630									
4631	016250	123453			CMPB	2(R4)+,2-(R3)			
4632	016252	001002			BNE	CMPB3			
4633	016254	123453			CMPB	2(R4)+,2-(R3)			
4634	016256	001401			BEQ	.+4			



```

4635 016260 104000
4636
4637
4638
4639 016262
4640 016262 000004
4641 016264 112737 000026 001202
4642 016272 000402
4643 016274 000000
4644 016276 000000
4645
4646 016300 013702 001550
4647 016304 010205
4648 016306 005065 016276
4649 016312 012762 000001 016274
4650 016320 056265 016274 016276 1$:
4651 016326 006362 016274
4652 016332 006362 016274
4653 016336 103370
4654 016340 022765 052525 016276
4655 016346 001401
4656 016350 104000
4657 016352 012762 177777 016274
4658 016360 046562 016276 016274
4659 016366 036265 016274 016276
4660 016374 001401
4661 016376 104000
4662 016400 006365 016276
4663 016404 026265 016274 016276
4664 016412 001401
4665
4666 016414 104000
4667 016416 000257
4668 016420 066265 016274 016276
4669 016426 103002
4670 016430 102001
4671 016432 100001
4672 016434 104000 ADD6:
4673
4674 016436 006362 016274
4675 016442 166265 016274 016276
4676 016450 103401
4677 016452 001401
4678 016454 104000 SUB6:
4679
4680 016456 112700 000377
4681 016462 010062 016274
4682 016466 012765 177777 016276
4683 016474 166500
4684 016500 001401
4685 016502 104000
4686 016504 066265 016274 016276 1$:
4687 016512 006362 016274
4688 016516 005162 016274
4689 016522 036265 016274 016276
4690 016530 001401

```

CMPB3: HLT  
:\*\*\*\*\*  
:TEST 26 CHECK BINARY OPS USING ADDRESS MODE 6  
:\*\*\*\*\*  
1\$T26:  
SCOPE  
MOVB #26,2\$TSTNM ;RESERVE TWO LOCATIONS  
BR .+6 ;RESERVED FOR SOURCE DATA  
SDATA: .WORD 0 ;RESERVED FOR DESTINATION DATA  
DDATA: .WORD 0  
MOV 2\$FACTOR,R2 ;GET RELOCATION FACTOR AND USE AS AN  
MOV R2,R5 ;INDEX VALUE TO POINT TO DATA  
CLR DDATA(5) ;PRESET DESTINATION DATA  
MOV #1,SDATA(2) ;THIS ROUTINE PUT A 1 BIT INTO EVERY  
1\$: BIS SDATA(2),DDATA(5) ;OTHER BIT POSITION IN THE DEST-  
ASL SDATA(2) ;INATION ADDRESS (52525)  
ASL SDATA(2)  
BCC 1\$  
CMP #52525,DDATA(5) ;CHECK RESULT  
BEQ .+4 ;ERROR! INCORRECT RESULT  
HLT  
MOV #-1,SDATA(2)  
BIC DDATA(5),SDATA(2) ;SOURCE DATA=125252  
BIT SDATA(2),DDATA(5)  
BEQ .+4  
HLT ;ERROR! BIT INST FAILED  
ASL DDATA(5) ;DDATA=125252  
CMP SDATA(2),DDATA(5)  
BEQ .+4  
HLT ;ERROR! CMP INST FAILED  
CCC  
ADD SDATA(2),DDATA(5)  
BCC ADD6  
BVC ADD6  
BPL .+4  
ADD6: HLT  
ASL SDATA(2) ;SDATA=52524  
SUB SDATA(2),DDATA(5)  
BCS SUB6  
BEQ .+4  
SUB6: HLT  
MOV #377,R0 ;R0=177777 (MOV B %R EXTENDS SIGN)  
MOV R0,SDATA(2)  
MOV #-1,DDATA(5)  
SUB DDATA(5),R0  
BEQ .+4  
HLT  
ADD SDATA(2),DDATA(5)  
ASL SDATA(2)  
COM SDATA(2)  
BIT SDATA(2),DDATA(5)  
BEQ .+4

4691	016532	104000		
4692	016534	005162	016274	
4693	016540	026265	016274	016276
4694	016546	001401		
4695	016550	104000		
4696	016552	026200	016274	
4697	016556	001352		

```

HLT
COM SDATA(2)
CMP SDATA(2),DDATA(5)
BEQ .+4
HLT
CMP SDATA(2),R0
BNE IS

```

```

*****
;TEST 27 CHECK BINARY BYTE OPS USING ADDRESS MODE 6
*****
TST27:

```

4701	016560			
4702	016560	000004		
4703	016562	112737	000027	001202
4704				
4705				
4706				

```

SCOPE
MOV #27,2#STSTNM
;NOTE: SDATA(2), AND DDATA(4) REFERENCE EVEN BYTE OF SOURCE & DEST DATA
;AND SDATA(3), AND DDATA(5) REFERENCE ODD BYTE OF SOURCE & DEST DATA

```

4707	016570	013702	001550	
4708	016574	010204		
4709	016576	010403		
4710	016600	005203		
4711	016602	010305		
4712	016604	000261		
4713	016606	012762	125252	016730
4714	016614	112763	177125	016730
4715	016622	016264	016730	016732
4716	016630	052764	125125	016732
4717	016636	136263	016730	016730
4718	016644	001401		
4719	016646	104000		

```

MOV 2#FACTOR,R2 ;GET INDEX VALUE
MOV R2,R4 ;R2 FOR SOURCE EVEN BYTE INDEX, R4 FOR
MOV R4,R3 ;DEST ODD BYTE, R3 FOR SOURCE EVEN
INC R3 ;AND R5 FOR DEST ODD BYTE
MOV R3,R5
SEC ;SET CARRY
MOV #125252,SDATAB(2)
MOV #177125,SDATAB(3) ;SOURCE DATA = 052652
MOV SDATA(2),DDATAB(4)
BIS #125125,DDATAB(4) ;DEST DATA = 177777
BITB SDATA(2),SDATAB(3)
BEQ .+4
BITB6: HLT

```

4720				
4721	016650	146264	016730	016732
4722	016656	103401		
4723	016660	104000		
4724	016662	126364	016730	016732
4725	016670	001401		
4726	016672	104000		

```

BICB SDATA(2),DDATAB(4)
BCS .+4
HLT ;ERROR MOV,BIS,BIT;BIC DO NOT AFFECT 'C'
CMPB SDATA(3),DDATAB(4)
BEQ .+4
HLT

```

4727				
4728	016674	146365	016730	016732
4729	016702	126265	016730	016732
4730	016710	001401		
4731	016712	104000		

```

BICB SDATA(3),DDATAB(5)
CMPB SDATA(2),DDATAB(5)
BEQ .+4
HLT

```

4732				
4733	016714	136564	016732	016732
4734	016722	001401		
4735	016724	104000		
4736	016726	000402		
4737	016730	000000		
4738	016732	000000		

```

BITB DDATA(5),DDATAB(4)
BEQ .+4
HLT
BR TST30 ;SKIP TO NEXT TEST
SDATAB: .WORD 0 ;RESERVED FOR SOURCE DATA
DDATAB: .WORD 0 ;RESERVED FOR DEST DATA

```

4739				
4740				
4741				
4742				
4743				
4744	016734			
4745	016734	000004		
4746	016736	112737	000030	001202

```

*****
;TEST 30 CHECK BINARY WORD OPS USING ADDRESS MODE 7
;R2=ADDRESS OF SOURCE DATA, AND R3= ADDRESS OF DEST DATA
*****
TST30:
SCOPE
MOV #30,2#STSTNM

```



4747	016744	000404			BR	UB7		;RESERVE FOUR WORDS
4748	016746	000000			SBIN7:	.WORD	0	;CONTAINS ADDRESS OF SOURCE DATA
4749	016750	000000			DBIN7:	.WORD	0	;CONTAINS ADDRESS OF DEST DATA
4750	016752	000000				.WORD	0	;CONTAINS SOURCE DATA
4751	016754	000000				.WORD	0	;CONTAINS DEST DATA
4752								
4753	016756	010700			UB7:	MOV	PC,R0	
4754	016760	024040				CMP	-(R0),-(R0)	
4755	016762	010002				MOV	R0,R2	
4756	016764	024242				CMP	-(R2),-(R2)	
4757	016766	010012				MOV	R0,(R2)	
4758	016770	010203				MOV	R2,R3	
4759	016772	024043				CMP	-(R0),-(R3)	
4760	016774	010013				MOV	R0,(R3)	
4761								
4762	016776	000261			SEC			
4763	017000	012777	100000	177740	MOV	#100000,SBIN7		;SOURCE DATA = 100000
4764	017006	017777	177734	177734	MOV	SBIN7,DBIN7		;DEST DATA = 100000
4765	017014	103001			BCC	MOV7		
4766	017016	100401			BMI	.+4		
4767	017020	104000			MOV7:	HLT		
4768	017022	006377	177722		ASL	DBIN7		;DEST DATA = 000000
4769	017026	102001			BVC	.+4		
4770	017030	001401			BEQ	.+4		
4771	017032	104000			HLT			
4772								
4773	017034	027777	177706	177706	CMP	SBIN7,DBIN7		;(R2)=100000,(R3)=000000
4774	017042	103402			BCS	CMP7		
4775	017044	102401			BVS	CMP7		
4776	017046	100401			BMI	.+4		
4777	017050	104000			CMP7:	HLT		
4778								
4779	017052	167777	177670	177670	SUB	SBIN7,DBIN7		;(R2)=100000,(R3)=100000
4780	017060	103003			BCC	SUB7		
4781	017062	102002			BVC	SUB7		
4782	017064	001401			BEQ	SUB7		
4783	017066	100401			BMI	.+4		
4784	017070	104000			SUB7:	HLT		
4785								
4786	017072	006277	177650		ASR	SBIN7		;(R2)=140000
4787	017076	067777	177644	177644	ADD	SBIN7,DBIN7		;(R2)=140000,(R3)=040000
4788	017104	103003			BCC	ADD7		
4789	017106	102002			BVC	ADD7		
4790	017110	001401			BEQ	ADD7		
4791	017112	100001			BPL	.+4		
4792	017114	104000			ADD7:	HLT		
4793								
4794	017116	047777	177624	177624	BIC	SBIN7,DBIN7		;(R2)=140000,(R3)=000000
4795	017124	001401			BEQ	.+4		
4796	017126	104000			HLT			
4797								
4798	017130	057777	177612	177612	BIS	SBIN7,DBIN7		;(R2)=140000,(R3)=140000
4799	017136	100401			BMI	.+4		
4800	017140	104000			HLT			
4801								
4802	017142	027777	177600	177600	CMP	SBIN7,DBIN7		



4803 017150 001401  
4804 017152 104000  
4805  
4806  
4807  
4808  
4809 017154  
4810 017154 000004  
4811 017156 112737 000031 001202  
4812 017164 005000  
4813 017166 005067 000072  
4814 017172 010707  
4815 017174 120707  
4816 017176 030707  
4817 017200 060007  
4818 017202 105707  
4819 017204 005507  
4820 017206 021007  
4821 017210 131007  
4822 017212 062707 000000  
4823 017216 023707 001550  
4824 017222 133707 001550  
4825 017226 000240  
4826  
4827  
4828 017230 163707 001550  
4829 017234 063707 001550  
4830 017240 000240  
4831 017242 024607  
4832 017244 132607  
4833 017246 026707 000012  
4834 017252 166707 000006  
4835 017256 046707 000002  
4836 017262 000401  
4837 017264 000000  
4838 017266 000004  
4839 017270 010702  
4840 017272 062702 000012  
4841 017276 012707 036574  
4842 017302 000000  
4843  
4844  
4845  
4846  
4847  
4848 017304  
4849 017304 012767 000001 162012  
4850 017312 000004  
4851 017314 112737 000032 001202  
4852  
4853  
4854  
4855 017322 112737 000032 001202  
4856 017330 010700  
4857 017332 005740  
4858 017334 010037 001554

```
BEQ .+4  
HLT  
:*****  
: *TEST 31 SOME MISCELLANEOUS OPERATIONS INVOLVING THE PC  
: * NOTE: NONE OF THESE OPERATIONS SHOULD AFFECT THE PC  
:*****  
TST31:  
SCOPE  
MOVB #31,@#STSTNM  
CLR RO  
CLR 1$  
MOV PC,PC  
CMPB PC,PC  
BIT PC,PC  
ADD RO,PC  
TSTB PC  
ADC PC  
CMP (RO),PC  
BITB (RO),PC  
ADD #0,PC  
CMP @#FACTOR,PC  
BITB @#FACTOR,PC  
NOP  
; THE NEXT TWO INSTRUCTION CAUSE THE PROGRAM TO JUMP TO THE UNRELOCATED  
; CODE AND TO RETURN ON THE FOLLOWING INST (IF THE CODE IS RELOCATED)  
SUB @#FACTOR,PC ; JUMPS TO UNRELOCATED CODE  
ADD @#FACTOR,PC ; RETURNS  
NOP  
CMP -(SP),PC  
BITB (SP)+,PC  
CMP 1$,PC  
SUB 1$,PC  
BIC 1$,PC  
BR .+4 ; BRANCH OVER 1$  
1$:  
0  
RELE2: SCOPE  
MOV PC,R2  
ADD #12,R2  
MOV #RELOC,PC ; GO RELOCATE PROGRAM CODE  
REL22: .WORD 0  
; 222222222222 LAST ADDRESS OF CODE TO BE RELOCATED 2222222222  
:*****  
: *TEST 32 CHECK BINARY BYTE OPS USING ADDRESS MODE 0  
:*****  
TST32:  
MOV #1,$TIMES ;; DO 1 ITERATION  
SCOPE  
MOVB #32,@#STSTNM  
.  
SBTTL START OF SECTION 3  
; 333333333333 FIRST ADDRESS TO BE RELOCATED 3333333333  
REL3: MOVB #STN-1,@#STSTNM  
MOV PC,RO ; GET PC  
TST -(RO) ; RO CONTAINS THE ADDRESS OF REL3  
MOV RO,@#FRSTAD ; SAVE
```

```

4859 017340 010700          MOV    PC,R0          ;GET CURRENT PC
4860 017342 162700 017342  SUB    #,R0          ;SUBTRACT RELOCATION FACTOR
4861 017346 010037 001550  MOV    R0,#FACTOR    ;SAVE RELOCATION FACTOR
4862 017352 010737 001212  MOV    PC,#SLPERR    ;SET LOOP ADDRESS
4863 017356 062737 000026 001212 ADD    #26,#SLPERR   ;ADJUST
4864 017364 013737 001212 001210 MOV    #SLPERR,#SLPADR
4865 017372 105737 001544  TSTB  #NEXEC        ;BR IF TEST CODE TO BE EXECUTED
4866 017376 001402          BEQ    .+6
4867 017400 000167 002230  JMP    RELE3
4868 017404 012783 125252  MOV    #125252,R3
4869 017410 010304          MOV    R3,R4          ;R3=R4=125252
4870 017412 140304          BICB  R3,R4          ;R3=125252,R4=125000
4871 017414 022704 125000  CMP    #125000,R4    ;CHECK RESULT
4872 017420 001401          BEQ    .+4
4873 017422 104000          HLT
4874
4875 017424 005004          CLR   R4              ;R3=125252,R4=0
4876 017426 150304          BISB  R3,R4          ;R3=125252,R4=000252
4877 017430 022704 000252  CMP    #252,R4       ;CHECK RESULT
4878 017434 001401          BEQ    .+4
4879 017436 104000          HLT
4880
4881 017440 110404          MOVB  R4,R4           ;R4=177652
4882 017442 022704 177652  CMP    #177652,R4    ;CHECK RESULT
4883 017446 001401          BEQ    .+4
4884 017450 104000          HLT
4885
4886 017452 132704 177525  BITB  #177525,R4
4887 017456 001401          BEQ    .+4
4888 017460 104000          HLT
4889
4890 017462 105104          COMB  R4              ;R4=177525
4891 017464 110404          MOVB  R4,R4           ;R4=000125
4892 017466 022704 000125  CMP    #125,R4       ;CHECK RESULT
4893 017472 001401          BEQ    .+4
4894 017474 104000          HLT
4895
4896 017476 150304          BISB  R3,R4          ;R3=125252,R4=000377
4897 017500 105204          INCB  R4
4898 017502 001401          BEQ    .+4
4899 017504 104000          HLT
4900
4901  ;*****
4902  ;*TEST 33 CHECK BINARY BYTE OPS USING ADDRESS MODE 7
4903  ;*****
4904  ;ST33:
4905  017506 000004          SCOPE
4906  017510 112737 000033 001202  MOVB  #33,#STSTNM
4907  017516 000406          BR    BINB7          ;RESERVE SPACE FOR ADDRESSES & DATA
4908  017520 000000          SBINB7: .WORD 0      ;CONTAINS ADDRESS OF SOURCE EVEN BYTE
4909  017522 000000          .WORD 0              ;CONTAINS ADDRESS OF SOURCE ODD BYTE
4910  017524 000000          .WORD 0              ;CONTAINS ADDRESS OF DEST EVEN BYTE
4911  017526 000000          .WORD 0              ;CONTAINS ADDRESS OF DEST ODD BYTE
4912  017530 000000          DBINB7: .WORD 0      ;CONTAINS SOURCE DATA
4913  017532 000000          .WORD 0              ;CONTAINS DEST DATA
4914  017534 010700          BINB7: MOV    PC,R0

```



4915	017536	024040			CMP	-(RO), -(RO)		;RO = ADDRESS OF DEST DATA
4916	017540	010060	177772		MOV	RO, -6(RO)		;LOAD ADDRESS OF DEST EVEN BYTE DATA
4917	017544	010060	177774		MOV	RO, -4(RO)		
4918	017550	005260	177774		INC	-4(RO)		;LOAD ADDRESS OF DEST ODD BYTE DATA
4919	017554	005740			TST	-(RO)		;RO=ADDRESS OF SOURCE DATA
4920	017556	010060	177770		MOV	RO, -10(RO)		;LOAD ADDRESS OF SOURCE EVEN BYTE DATA
4921	017562	010060	177772		MOV	RO, -6(RO)		
4922	017566	005260	177772		INC	-6(RO)		;LOAD ADDRESS OF SOURCE ODD BYTE DATA
4923								
4924	017572	005002			CLR	R2		;SET INDEX REGISTERS
4925	017574	012703	000002		MOV	#2, R3		;JSBINB7(2); JSBINB7(3) REFERENCE EVEN &
4926	017600	012704	177774		MOV	#-4, R4		;ODD BYTE SOURCE DATA; JSBINB7(4); JSBINB7(5)
4927	017604	012705	177776		MOV	#-2, R5		;REFERENCE DEST EVEN& ODD BYTE DATA
4928								
4929								
4930	017610	005020			CLR	(RO)+		;PRESET SOURCE DATA
4931	017612	005010			CLR	(RO)		;PRESET DEST DATA
4932	017614	013746	001550		MOV	#FACTOR, -(SP)		;GET RELOCATION FACTOR
4933	017620	061602			ADD	(SP), R2		;AND ADD TO INDEX VALUES
4934	017622	061603			ADD	(SP), R3		
4935	017624	061604			ADD	(SP), R4		
4936	017626	062605			ADD	(SP)+, R5		
4937								
4938	017630	112773	177777	017520	MOVB	#-1, JSBINB7(3)		;SRC DATA = 177400
4939	017636	132772	000377	017520	BITB	#377, JSBINB7(2)		;CHECK THAT EVEN BYTE WAS NOT AFFECTED
4940	017644	001401			BEQ	.+4		;BY MOVB INSTRUCTION
4941	017646	104000			HLT			
4942								
4943	017650	157374	017520	017530	BISB	JSBINB7(3), JSBINB7(4)		
4944	017656	105274	017530		INCB	JSBINB7(4)		;CHECK THAT BIS SET ALL BITS
4945	017662	001401			BEQ	.+4		
4946	017664	104000			HLT			
4947								
4948	017666	105375	017530		DECB	JSBINB7(5)		;DEST DATA = 177400
4949	017672	005274	017530		INC	JSBINB7(4)		;DEST DATA = 177401
4950	017676	127375	017520	017530	CMPB	JSBINB7(3), JSBINB7(5)		
4951	017704	001401			BEQ	.+4		
4952	017706	104000			HLT			
4953								
4954	017710	147375	017520	017530	BICB	JSBINB7(3), JSBINB7(5)		
4955	017716	001401			BEQ	.+4		
4956	017720	104000			HLT			
4957								
4958	017722	105073	017520		CLRB	JSBINB7(3)		;SRC DATA = 000000
4959								
4960								
4961	017726	157473	017530	017520	BISB	JSBINB7(4), JSBINB7(3)		
4962	017734	106174	017530		ROLB	JSBINB7(4)		
4963	017740	103372			BCC	BIS7		
4964	017742	022772	177400	017520	CMP	#177400, JSBINB7(2)		;CHECK RESULT
4965	017750	001401			BEQ	.+4		
4966	017752	104000			HLT			
4967								
4968	017754	000372	017520		SWAB	JSBINB7(2)		;SRC DATA = 000377
4969	017760	112775	000200	017530	MOVB	#200, JSBINB7(5)		;DEST DATA = 100000
4970								

; THIS ROUTINE SETS ALL BITS IN THE SOURCE ODD BYTE BY BISING A BIT FROM  
; THE DEST EVEN BYTE INTO THE SOURCE ODD BYTE

BIS7:





I09

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 112  
T33 CHECK BINARY BYTE OPS USING ADDRESS MODE 7

4989 020054 000277

SCC

;SET ALL CC'S

4990	020056	160212				SUB	R2,(R2)		
4991	020060	104000				HLT			
4992	020062	060222				ADD	R2,(R2)+		
4993	020064	104000				HLT			
4994	020066	006342				ASL	-(R2)		
4995	020070	104000				HLT			
4996	020072	106512				MFPD	(R2)		
4997	020074	104000				HLT			
4998	020076	170412				CLRF	(R2)		
4999	020100	104000				HLT			
5000	020102	042202				BIC	(R2)+,R2		
5001	020104	104000				HLT			
5002	020106	164202				SUB	-(R2),R2		
5003	020110	104000				HLT			
5004	020112	155202				BISB	2-(R2),R2		
5005	020114	104000				HLT			
5006	020116	105532				ADCB	2(R2)+		
5007	020120	104000				HLT			
5008	020122	163302				SUB	2(R3)+,R2		
5009	020124	104000				HLT			
5010	020126	005733				TST	2(R3)+		
5011	020130	104000				HLT			
5012	020132	106533				MFPD	2(R3)+		
5013	020134	104000				HLT			
5014	020136	170453				CLRD	2-(R3)		
5015	020140	104000				HLT			
5016	020142	137702	177775			BITB	2.+1,R2		
5017	020146	104000				HLT			
5018	020150	105477	177773			NEGB	2.-1		
5019	020154	104000				HLT			
5020	020156	000406				BR	25		
5021									
5022	020160	062716	000002		15:	ADD	2,(SP)	;ADJUST RETURN PC	
5023	020164	052766	000017	000002		BIS	217,2(SP)	;SET CONDITION CODES ON RETURN	
5024	020172	000002				RTI			
5025									
5026	020174	012706	000700		25:	MOV	2USESTK,SP	;RESET STACK PTR	
5027	020200	012737	055064	000004		MOV	2ERPRT,2ERRVEC	;RESET TIME OUT VECTOR	
5028	020206	012737	054774	000010		MOV	2RESERR,2RESVEC		
5029									
5030									
5031									
5032	020214								
5033	020214	000004				SCOPE			
5034	020216	112737	000034	001202		MOVB	234,22STSTNM		
5035	020224	010700				MOV	PC,R0		
5036	020226	062700	000012			ADD	212,R0	;SET ADDRESS FOR JMP INST	
5037	020232	000277				SCC		;SET CC'S	
5038	020234	000110				JMP	(R0)		
5039	020236	000402				BR	2.+6		
5040	020240	000250				CLN		;JMP INST JUMPS HERE	
5041	020242	000775				BR	2.-4		
5042									
5043	020244	103003				BCC	JMP1		
5044	020246	102002				BVC	JMP1		
5045	020250	001001				BNE	JMP1		



```

5046 020252 100001
5047 020254 104000      JMP1:  BPL      .+4
5048
5049 020256 005002      CLR      R2      ;SET INDICATOR
5050 020260 010703      MOV     PC,R3
5051 020262 000401      BR      .+4      ;RESERVE WORD FOR JMP ADDRESS
5052 020264 000000      .WORD   0      ;CONTAINS ADDRESS FOR JMP INST
5053 020266 005723      TST     (R3)+
5054 020270 010313      MOV     R3,(R3)
5055 020272 010300      MOV     R3,R0
5056 020274 062713 000022      ADD     #22,(R3) ;(R3) IS JMP ADDRESS
5057 020300 010300      MOV     R3,R0
5058 020302 000133      JMP     @R3+     ;JUMP TO ADDRESS CONTAINED IN R3
5059 020304 000402      BR      .+6
5060 020306 005102      COM     R2      ;COMPLEMENT INDICATOR
5061 020310 000775      BR      -4
5062 020312 005202      INC     R2      ;CHECK INDICATOR
5063 020314 001003      BNE     JMP3
5064 020316 005720      TST     (R0)+
5065 020320 020003      CMP     R0,R3   ;CHECK AUTO-INC R3
5066 020322 001401      BEQ
5067 020324 104000      JMP3:  HLT
5068
5069 020326 005002      CLR     R2      ;SET INDICATOR
5070 020330 010704      MOV     PC,R4
5071 020332 010400      MOV     R4,R0
5072 020334 000402      BR      1$
5073 020336 005102      COM     R2      ;COMPLEMENT INDICATOR
5074 020340 000403      BR      2$
5075 020342 022424      1$:    CMP     (R4)+,(R4)+ ;R4=JMP ADDRESS
5076 020344 005724      TST     (R4)+   ;USE R4 AS ADDRESS
5077 020346 000144      JMP     -(R4)   ;CHECK INDICATOR
5078 020350 005202      2$:    INC     R2
5079 020352 001003      BNE     JMP4
5080 020354 022020      CMP     (R0)+,(R0)+ ;CHECK AUTO-DEC R4
5081 020356 020004      CMP     R0,R4
5082 020360 001401      BEQ
5083 020362 104000      JMP4:  HLT
5084
5085 020364 010703      MOV     PC,R3
5086 020366 000401      BR      .+4      ;RESERVE WORD FOR JMP ADDRESS
5087 020370 000000      1$:    .WORD   0      ;CONTAINS JUMP ADDRESS
5088 020372 005723      TST     (R3)+
5089 020374 010313      MOV     R3,(R3)
5090 020376 062723 000016      ADD     #16,(R3)+ ;LOAD CHECK REGISTER
5091 020402 010300      MOV     R3,R0
5092 020404 000402      BR      3$
5093 020406 005102      2$:    COM     R2
5094 020410 000401      BR      4$
5095 020412 000153      3$:    JMP     @-(R3)   ;JUMP TO 2$ VIA 1$ ABOVE
5096 020414 005202      4$:    INC     R2      ;CHECK INDICATOR
5097 020416 001003      BNE     JMP5
5098 020420 005740      TST     -(R0)
5099 020422 020003      CMP     R0,R3   ;CHECK AUTO-DEC R3
5100 020424 001401      BEQ
5101 020426 104000      JMP5:  HLT

```

```

5102
5103 020430 000402
5104 020432 005102 1S: BR 2S
5105 020434 000402 BR R2 ;COMPLEMENT INDICATOR
5106 020436 000167 177770 2S: JMP 3S
5107 020442 005202 3S: INC R2
5108 020444 001401 BEQ .+4
5109 020446 104000 JMP6: HLT
5110
5111 020450 012767 020466 000020 MOV #1S,7S ;SET UP JMP ADDRESS
5112 020456 063767 001550 000012 ADD #FACTOR,7S ;ADD RELOCATION FACTOR
5113 020464 000402 BR 2S ;GO TO JMP 27S INST
5114 020466 005102 1S: COM R2 ;COMPLEMENT INDICATOR
5115 020470 000403 BR 3S ;GO TO CHECK ROUTINE
5116 020472 000177 000000 2S: JMP 27S ;JMP TO 1S ABOVE VIA 7S
5117 020476 000000 7S: .WORD 0 ;CONTAINS JMP ADDRESS
5118 020500 005202 3S: INC R2 ;CHECK INDICATOR
5119 020502 001401 BEQ .+4
5120 020504 104000 JMP7: HLT
5121
5122 ;*****
5123 ;#TEST 3S CHECK JSR INSTRUCTIONS
5124 ;*****
5125 020506 000004 TST3S:
5126 020510 112737 000035 001202 SCOPE
5127 020516 013705 001550 JSR1: MOVB #3S,2#STSTNM
5128 020522 012702 020554 JSR1: MOV #3S,R2 ;GET RELOCATION FACTOR
5129 020526 060502 ADD R5,R2 ;FORM DEST ADRS
5130 020530 000277 SCC ;ADD RELOCATION FACTOR
5131 020532 000242 CLV ;PRESET CC'S
5132 020534 004512 JSR R5,(R2) ;GO TO 3S VIA R2
5133 020536 005702 1S: TST R2 ;CHECK INDICATOR
5134 020540 001017 BNE 4S ;R2 SHOULD=0
5135 020542 023705 001550 CMP #FACTOR,R5 ;CHECK THAT RTS R5 RESTORED R5
5136 020546 001014 BNE 4S
5137 020550 000414 BR JSR3 ;GO TO NEXT TEST
5138 020552 000205 2S: RTS R5 ;RETURN FROM SUBROUTINE
5139 020554 103011 3S: BCC 4S ;CHECK THAT JSR DID NOT
5140 020556 102410 BVS 4S
5141 020560 001007 BNE 4S ;AFFECT CC'S
5142 020562 100006 BPL 4S
5143 020564 005002 CLR R2 ;CLEAR INDICATOR
5144 020566 012704 020536 MOV #1S,R4 ;GET UNRELOCATED RETURN ADDRESS
5145 020572 061604 ADD (SP),R4 ;ADD RELOCATION FACTOR (OLD R5)
5146 020574 020405 CMP R4,R5 ;CHECK THAT OLD R5 WAS PLACED ON THE
5147 020576 001765 BEQ 2S ;STACK & THAT NEW R5 CONTAINS RETURN PC
5148 020600 104000 4S: HLT ;ERROR! ABOVE
5149
5150 ;CHECK JSR INSTRUCTION ADDRESS MODE 3
5151 020602 013704 001550 JSR3: MOV #FACTOR,R4 ;GET RELOCATION FACTOR
5152 020604 005000 CLR R0 ;SET INDICATOR
5153 020610 012705 020630 MOV #1S,R5
5154 020614 060405 ADD R4,R5 ;SET UP JSR DEFERRED ADRS
5155 020616 010502 MOV R5,R2
5156 020620 012715 020646 MOV #5S,(R5)
5157 020624 060415 ADD R4,(R5) ;(R5)=DEST ADRS

```



```

5158 020626 000401          BR      2$          ;RESERVE WORD FOR ADDRESS
5159 020630 000000          1$: .WORD 0          ;CONTAINS DEST ADRS FOR JSR
5160 020632 004435          2$: JSR  R4,2(R5)+   ;JSR TO 5$ VIA 1$ ABOVE
5161 020634 005200          3$: INC  R0          ;CHECK INDICATOR
5162 020636 001013          BNE    6$
5163 020640 000413          BR     JSR4
5164 020642 005100          4$: COM  R0          ;COMPLEMENT INDICATOR
5165 020644 000204          RTS    4          ;RETURN FROM SUBROUTINE
5166 020646 012703 020634  5$: MOV  #3$,R3      ;GET UNRELOCATED RETURN ADDRESS
5167 020652 061603          ADD   (SP),R3    ;ADD RELOCATION FACTOR (OLD R4)
5168 020654 020403          CMP   R4,R3
5169 020656 001003          BNE   6$
5170 020660 005722          TST  (R2)+
5171 020662 020205          CMP  R2,R5      ;CHECK AUTO-INC R5
5172 020664 001766          BEQ  4$        ;GO TO RTS
5173 020666 104000          6$: HLT          ;ERROR ABOVE
5174
5175          ;CHECK JSR INST ADDRESS MODE 4
5176 020670 013704 001550  JSR4: MOV  #FACTOR,R4
5177 020674 010405          MOV  R4,R5
5178 020676 010703          MOV  PC,R3
5179 020700 000401          BR   2$
5180 020702 000405          1$: BR   4$
5181 020704 022323          2$: CMP  (R3)+,(R3)+
5182 020706 000277          SCC
5183 020710 004443          JSR  R4,-(R3)   ;GO TO 2$
5184 020712 104000          3$: HLT
5185 020714 000414          BR   JSR6      ;GO TO NEXT TEST
5186 020716 103012          4$: BCC  5$
5187 020720 102011          BVC  5$
5188 020722 001010          BNE  5$
5189 020724 100007          BPL  5$
5190 020726 012702 020712  MOV  #3$,R2      ;GET UNRELOCATED RETURN ADDRESS
5191 020732 061602          ADD  (SP),R2    ;ADD RELOCATION FACTOR (OLD R4)
5192 020734 020204          CMP  R2,R4      ;CHECK THAT CALCULATED RETURN
5193 020736 001002          BNE  5$        ;PC = NEW R4
5194 020740 005724          TST  (R4)+
5195 020742 000204          RTS  R4
5196 020744 104000          5$: HLT
5197
5198          ;TEST JSR INST ADDRESS MODE 6
5199 020746 000401  JSR6: BR      2$
5200 020750 000405          1$: BR      3$
5201 020752 010700          2$: MOV  PC,R0
5202 020754 004767 177770  JSR  PC,1$
5203 020760 100407          BMI  JSR7      ;GO TO NEXT TEST
5204 020762 104000          HLT          ;ERROR ON CC'S
5205 020764 022020          3$: CMP  (R0)+,(R0)+
5206 020766 020016          CMP  R0,(SP)   ;CHECK THAT RETURN ADDRESS IS ON THE
5207 020770 001401          BEQ  .+4       ;STACK
5208 020772 104000          HLT
5209 020774 000270          SEN
5210 020776 000207          RTS  PC      ;SET N
5211
5212          ;TEST JSR INST ADDRESS MODE 7
5213 021000 013746 001550  JSR7: MOV  #FACTOR,-(SP) ;GET RELOCATION FACTOR

```



5214 021004 062716 021024  
5215 021010 000277  
5216 021012 004076 000000  
5217 021016 003003  
5218 021020 102002  
5219 021022 000402  
5220  
5221 021024 000200  
5222 021026 104000  
5223 021030  
5224  
5225  
5226  
5227  
5228  
5229 021030  
5230 021030 000004  
5231 021032 112737 000036 001202  
5232 021040 012705 000022  
5233 021044 005000  
5234 021046 052740 000200  
5235 021052 011015  
5236 021054 011504  
5237 021056 010746  
5238 021060 062716 000036  
5239 021064 012645  
5240 021066 042710 000357  
5241 021072 052710 000244  
5242 021076 012003  
5243 021100 010340  
5244 021102 000004  
5245 021104 012737 046720 000020 10\$:  
5246 021112 104000  
5247 021114 000447  
5248  
5249 021116 012002 1\$:  
5250  
5251 021120 012725 046720  
5252 021124 012715 000200  
5253 021130 010746  
5254 021132 062716 177752  
5255 021136 022626  
5256 021140 001026  
5257 021142 022603  
5258 021144 001024  
5259 021146 032703 140000  
5260 021152 100403  
5261 021154 020204  
5262 021156 001017  
5263 021160 000404  
5264  
5265  
5266 021162 052704 030000 3\$:  
5267 021166 020204  
5268 021170 001012  
5269

```

ADD #1$, (SP) ;FORM ADDRESS OF 1$ BELOW
SCC ;SET ALL CC'S
JSR R0, 2(SP) ;JSR TO 1$
BGT 3$
BVC 3$
BR 4$

1$: RTS R0 ;RETURN
3$: HLT ;ERROR!! INCORRECT CC'S
4$:

*****
*TEST 36 CHECK IOT TRAP (AND ROLB/ASLB)
* THIS TEST CHECKS THAT THE PSW IS CORRECT AFTER THE IOT AND THAT THE
* 'NEW' PSW (FROM IOTVEC+2) IS CORRECT.
*****
TST36:
SCOPE
MOV #36, 2*STSTNM
IOTTST: MOV #IOTVEC+2, R5
CLR R0
BIS #PR4, -(R0) ;SET PRIORITY LEVEL 4 IN PSW
MOV (R0), (R5) ;SET IOTVEC+2 = PSW
MOV (R5), R4 ;SAVE IN R4
MOV PC, -(SP)
ADD #1$, (SP)
MOV (SP)+, -(R5) ;LOAD IOT TRAP VECTOR
BIC #PR7+17, (R0) ;PSW=X XXX X00 101 1X1000
BIS #PR5+4, (R0) ;R3 = PSW ABOVE
MOV (R0)+, R3 ;RESTORE PSW (MOV CHANGED IT)
MOV R3, -(R0)
IOT
MOV #SCOPE, 2*IOTVEC ;RESTORE IOT VECTOR
HLT ;ERROR! IOT FAILED TO TRAP
BR TST37 ;;GO TO NEXT TEST

1$: MOV (R0)+, R2 ;GET PSW AFTER IOT TRAP
;NOTE: R0=0
MOV #SCOPE, (R5)+ ;RESTORE IOTVEC
MOV #PR4, (R5) ;AND IOTVEC+2
MOV PC, -(SP) ;FORM PC OF 10$ ABOVE
ADD #10$, -(SP)
CMP (SP)+, (SP)+ ;CHECK RETURN PC ON STACK
BNE 99$
CMP (SP)+, R3 ;CHECK SAVED PSW
BNE 99$
BIT #UM, R3 ;BRANCH TO 3$ IF IN USER MODE
BMI 3$
CMP R2, R4 ;CHECK PSW AFTER IOT
BNE 99$
BR 4$

3$: BIS #PUM, R4 ;SET PREV USER MODE
CMP R2, R4 ;CHECK PSW AFTER IOT
BNE 99$

```

```

5270 021172 005002      4S:  CLR      R2
5271 021174 000261      SEC
5272 021176 106100      ROLB     RO      ; ROTATE RO
5273 021200 102376      BVC     .-2      ; UNTIL V SETS (RO=200)
5274
5275 021202 106300      ASLB     RO      ; SHIFT SHOULD SET CARRY
5276 021204 103004      BCC     99S
5277 021206 102003      BVC     99S
5278 021210 001002      BNE     99S
5279 021212 005700      TST     RO
5280 021214 001401      BEQ     .+4
5281 021216 104000      99S:  HLT
5282
5283
5284 021220 042704 000340      BIC     #PR7,R4
5285 021224 010437 177776      MOV     R4,#PSW      ; RESTORE PSW
5286 021230 012706 000700      MOV     #USESTK,SP   ; RESTORE STACK PTR
5287
5288
5289
5290
5291 021234 000004
5292 021236 112737 000037 001202
5293
5294 021244 012737 047160 000020
5295 021252 012737 000340 000022
5296 021260 005000
5297 021262 010746
5298 021264 062716 000030
5299 021270 012637 000030
5300 021274 000262
5301 021276 013737 177776 000032
5302 021304 000265
5303 021306 104000
5304 021310 001433
5305 021312 000004
5306 021314 102027      EMT1:  BVC     EMT1B
5307 021316 105100      COMB    RO
5308 021320 105500      ADCB   RO
5309 021322 106000      RORB   RO
5310 021324 102023      BVC     EMT1B
5311 021326 100022      BPL     EMT1B
5312 021330 000257      CCC
5313 021332 105400      NEGB   RO      ; RO=000200,CC'S=1010
5314 021334 102017      BVC     EMT1B
5315 021336 100016      BPL     EMT1B
5316 021340 000242      CLV
5317 021342 000261      SEC
5318 021344 105300      DECB   RO      ; CLEAR 'V'
5319 021346 102012      BVC     EMT1B      ; AND SET 'C'
5320 021350 100411      BMI     EMT1B      ; RO=000177,CC'S=0011
5321 021352 000242      CLV
5322 021354 105200      INCB   RO      ; CLEAR 'V'
5323 021356 103006      BCC     EMT1B      ; RO=000200,CC'S=1011
5324 021360 102005      BVC     EMT1B
5325 021362 100004      BPL     EMT1B

```

```

*****
:TEST 37 CHECK EMT TRAP SEQUENCE
*****

```

TST37:

EMT1:



```

5326 021364 000242          CLV          ;CLEAR 'V'
5327 021366 106200          ASRB        RO          ;SHIFT RO UNTIL 'V' CLEARS
5328 021370 102776          BVS        .-2
5329 021372 000401          BR         .+4
5330 021374 000004          EMT1B:    HLT
5331 021376 000002          RTI          ;ERROR!
5332 021400 105500          EMT1C:    ADCB        RO          ;EXIT WITH RO=000377
5333 021402 103003          BCC        EMT1D       ;RO=000000
5334 021404 001002          BNE        EMT1D
5335 021406 005700          TST        RO
5336 021410 001401          BEQ        .+4
5337 021412 000004          EMT1D:    HLT
5338 021414 012737 047160 000030      MOV        #ERROR, @EMTVEC ;RESTORE EMT TO ERROR
5339 021422 012737 000340 000032      MOV        #PR7, @EMTVEC+2 ;SET PRIORITY 7 ON ERROR
5340 021430 012737 046720 000020      MOV        #SCOPE, @IOTVEC ;RESTORE IOT VECTOR
5341 021436 005037 000022      CLR        @IOTVEC+2
5342          .EQUIV ERROR, HLT ;REDEFINE HLT CALL
5343          ;*****
5344          ;*TEST 40 CHECK TRAP INSTRUCTION TRAP SEQUENCE
5345          ;*****
5346          †ST40:
5347 021442 000004          SCOPE
5348 021444 112737 000040 001202      MOVB       #40, @STSTNM
5349 021452 052737 000340 177776      BIS        #PR7, @PSW ;LOCK OUT LINE CLOCK
5350 021460 052737 000340 000016      BIS        #PR7, @TBITVEC+2
5351 021466 010746          MOV        PC, -(SP)
5352 021470 062716 000056          ADD        #TRAP1-, (SP)
5353 021474 012637 000034          MOV        (SP)+, @TRAPVEC
5354 021500 000270          SEN
5355 021502 013737 177776 000036      MOV        @PSW, @TRAPVEC+2 ;SET N
5356 021510 000261          SEC        ;RETAIN CURRENT PSW ON TRAP
5357 021512 010700          MOV        PC, RO ;SET CARRY
5358 021514 000264          SEZ
5359 021516 104400          TRAP      ;SET Z BIT
5360 021520 103404          BCS        .+12 ;TRAP TO TRAP1
5361 021522 012737 053566 000034      MOV        #STRAP, @TRAPVEC ;RESTORE TRAP VECTOR
5362 021530 104000          HLT
5363 021532 001404          BEQ        .+12
5364 021534 012737 053566 000034      MOV        #STRAP, @TRAPVEC ;RESTORE TRAP VECTOR
5365 021542 104000          HLT
5366 021544 000420          TRAP1:   BR         TRAP1C
5367 021546 100404          BMI        .+12 ;N BIT GOT SET ON TRAP
5368 021550 012737 053566 000034      MOV        #STRAP, @TRAPVEC ;RESTORE TRAP VECTOR
5369 021556 104000          HLT
5370 021560 062700 000004          ADD        #4, RO
5371 021564 020016          CMP        RO, (SP) ;CHECK LOW BYTE OF RETURN PC ON
5372 021566 001404          BEQ        .+12 ;STACK
5373 021570 012737 053566 000034      MOV        #STRAP, @TRAPVEC ;RESTORE TRAP VECTOR
5374 021576 104000          HLT
5375 021600 124646          CMPB      -(SP), -(SP)
5376 021602 032626          BIT        (SP)+, (SP)+
5377 021604 000002          RTI          ;RETURN TO INST FOLLOWING TRAP (1$)
5378
5379 021606 012702 000036      TRAP1C:  MOV        #TRAPVEC+2, R2 ;RESTORE VECTORS
5380 021612 012712 000340          MOV        #PR7, (R2)
5381 021616 012742 053566          MOV        #STRAP, -(R2)

```





```

5438 022064 021666 177776      CMP      (SP),-2(SP)      ;SO IS COMPARE
5439 022070 012656              MOV      (SP)+,2-(SP)   ;BECAUSE OF ADDRESS MODE 5
5440 022072 057636 000000      BIS      2(SP),2(SP)+   ;BECAUSE OF ADDRESS MODE 3
5441 022076 054676 000000      BIS      -(SP),2(SP)   ;BECAUSE OF ADDRESS MODE 7
5442 022102 005006              CLR      SP
5443 022104 013766 020000 020000  MOV      2#20000,20000(SP)
5444 022112 000423              BR       3$            ;BRANCH OVER NON KERNEL MODE TESTS
5445
5446
5447 022114 156737 000171 177777 ;NOTE: NO OVERFLOW TRAP WILL OCCUR IF NOT IN KERNEL MODE!!!
5448 022122 012706 000376 1$:    BISB     7$+1,2#PSW+1   ;RESTORE MODE BITS IN PSW
5449 022126 016646 177776      MOV      2#376,SP      ;SET STACK PTR
5450 022132 051616 177776      MOV      -2(SP),-(SP)  ;SHOULD NOT TRAP
5451 022134 061666 177776      BIS      (SP),(SP)
5452 022140 105037 177777      ADD      (SP),-2(SP)
5453 022144 000451              CLRB    2#PSW+1       ;SET KERNEL MODE
5454
5455
5456 022146 012600              ;ERROR SERVICE ROUTINE
5457 022150 012602 2$:    MOV      (SP)+,R0      ;SAVE PC OF INSTRUCTION THAT TRAPPED
5458 022152 012706 000700      MOV      (SP)+,R2      ;SAVE PSW
5459 022156 104000              MOV      2#USESTK,SP   ;SET STACK PTR
5460
5461
5462 022160 000443              HLT
5463
5464
5465 022162 062737 000066 000004 ;RO CONTAINS PC, R2 CONTAINS PSW
5466 022170 010306 3$:    BR       6$            ;EXIT TEST
5467 022172 112702 000001      ;THE BELOW INSTRUCTIONS WILL CAUSE A STACK OVERFLOW
5468 022176 005000              ;STACK PTR IS AT 376
5469 022200 005016 3$:    ADD      2#4$-2$,2#ERRVEC ;SET ERROR VECTOR TO 4$
5470 022202 006302              MOV      R3,SP         ;SET STACK PTR AT 376
5471 022204 105226              MOVB    #1,R2
5472 022206 006302              CLR     R0
5473 022210 060746              CLR     (SP)           ;SETS BIT 0 IN R0
5474 022212 006302              ASL     R2             ;SHIFT INDICATOR BIT
5475 022214 000003              INCB   (SP)+          ;SETS BIT 1 IN R0
5476 022216 006302              ASL     R2
5477 022220 004767 000014      ADD     PC,-(SP)       ;SETS BIT 2 IN R0
5478 022224 006302              ASL     R2
5479 022226 050666 177776      BPT    R2             ;SETS BIT 3 IN R0
5480 022232 000410              ASL     R2
5481
5482
5483 022234 050200 4$:    JSR     PC,40$        ;SETS BIT 4 IN R0
5484 022236 000002              ASL     R2
5485
5486 022240 052700 001000 40$:   BIS     #1000,R0       ;SET IND THAT JSR WAS EXECUTED
5487 022244 000207              RTS     PC
5488
5489 022246 052700 000400 41$:   BIS     #400,R0       ;SET IND THAT BPT WAS EXECUTED
5490 022252 000002              RTI
5491
5492
5493 022254 012706 000700 5$:    MOV      2#USESTK,SP   ;CHECK THAT ABOVE INSTRUCTIONS DID TRAP ;SET STACK PTR

```



```

5494 022260 022700 001477          CMP      #1477,R0          ;EACH INSTRUCTION SET A BIT IN R0
5495 022264 001401          BEQ      .+4            ;RO= 1477
5496 022266 104000          HLT
5497
5498
5499 022270 012706 001200          ;EXIT ROUTINE
5500 022274 016737 177522 000014 6S:      MOV      #KERSTK,SP      ;SET KERNEL STACK PTR
5501 022302 005037 000016          MOV      43S,2#BPTVEC   ;RESTORE BPT VECTOR
5502 022306 012746          CLR      2#BPTVEC+2
5503 022310 000000          MOV      (PC)+,-(SP)    ;PUSH OLD PSW ONTO STACK
5504 022312 010746          7S:      .WORD      0      ;CONTAINS SAVED PSW
5505 022314 062716 000006          MOV      PC,-(SP)      ;PUSH CURRENT PC ONTO STACK
5506 022320 000002          ADD      #6,(SP)       ;ADD OFFSET
5507 022322 012706 000700          RTI
5508 022326 012737 055064 000004          MOV      #USESTK,SP     ;SET STACK PTR
5509 022334 013737 177776 000006          MOV      #ERPRT,2#ERRVEC ;RESET TIME OUT VECTOR
5510 022342 052737 000340 000006          MOV      2#PSW,2#ERRVEC+2
5511
5512          BIS      #PR7,2#ERRVEC+2
5513          ;*****
5514          ;#TEST 42 CHECK THAT ALL RESERVED INSTRUCTIONS TRAP
5515          ;*****
5516          ;T42:
5517          SCOPE
5518          MOV      #42,2#STSTNM
5519          RESTRP: MOV      #5S,R2          ;GET ADDRESS OF RESERVED INSTRUCTION TABLE
5520          ADD      2#FACTOR,R2
5521          MOV      #4S,2#RESVEC   ;SET RESERVED INSTRUCTION TRAP
5522          ADD      2#FACTOR,2#RESVEC
5523          1S:      MOV      (R2)+,R3      ;GET FIRST RESERVED INSTRUCTION
5524          BEQ      7S            ;0 TERMINATES THE TABLE
5525          MOV      (R2)+,R4      ;GET LAST RESERVED INSTRUCTION IN GROUP
5526          2S:      MOV      R3,(PC)    ;EXECUTE RESERVED INSTRUCTION
5527          3S:      .WORD      0      ;CONTAINS RESERVED INSTRUCTION
5528          HLT          ;ERROR! INSTRUCTION IN R3
5529          HLT          ;(2S) ABOVE FAILED TO CAUSE A
5530          HLT          ;RESERVED INSTRUCTION TRAP
5531          BR      41S
5532          4S:      MOV      #41S,(SP)   ;ADJUST RETURN PC
5533          ADD      2#FACTOR,(SP)    ;TO RETURN TO 41S
5534          RTI          ;RETURN TO 41S
5535          41S:     CMP      R3,R4      ;HAS GROUP OF RESERVED INSTRUCTIONS
5536          BEQ      1S            ;BEEN EXECUTED
5537          INC      R3            ;INCREMENT THIS RESERVED INSTRUCTION
5538          BR      2S            ;TO NEXT ONE AND EXECUTE
5539          ;TABLE OF 11/6X RESERVED INSTRUCTIONS (0 TERMINATES THE TABLE)
5540          5S:      7          ;GROUP 1
5541          77
5542          210        ;GROUP 2
5543          237
5544          7000       ;GROUP 3
5545          7777
5546          75040     ;GROUP 4
5547          76577
5548          ;MED INST. = 076600
5549          ;GROUP 5
5550          ;XFC INSTS. = 0767XX
    
```



# G10

```

5550                                     ;(76700-76777)
5551 022474 106400                       106400      ;GROUP 6
5552 022476 106477                       106477      ;
5553 022500 106700                       106700      ;GROUP 7
5554 022502 107777                       107777      ;
5555 022504 000000                       0           ;0 TERMINATES THE TABLE
5556
5557 022506 012737 054774 000010 7$:   MOV      #RESERR,#RESVEC      ;RESTORE RESERVED TRAP
5558                                     ;*****
5559                                     ;*TEST 43 CHECK THAT ALL BITS IN THE PSW CAN BE SET AND CLEARED
5560                                     ;*****
5561 022514                                     †ST43:
5562 022514 000004                                     SCOPE
5563 022516 112737 000043 001202         MOVB     #43,#STSTNM
5564 022524 105737 001545         PSWCHK: TSTB     #MNON      ;IF MEM MGMT IS ON SKIP THIS TEST
5565 022530 001103                                     BNE     4$
5566 022532 013767 177776 000166         MOV     #PSW,3$          ;SAVE STATUS
5567 022540 005037 177776         CLR     #PSW            ;CLEAR MODE BITS IN PSW
5568 022544 004737 054112         JSR    PC,#CLRTRBIT     ;GO CLEAR 'T' BIT IF SET
5569 022550 013746 000016         MOV     #TBITVEC+2,-(SP)
5570 022554 012704 177776         MOV     #PSW,R4         ;LOAD ADDRESS OF PSW INTO R4
5571 022560 000250                                     CLN
5572 022562 005714                                     TST    (R4)             ;CHECK THAT PSW WAS CLEARED
5573 022564 001401                                     BEQ    .+4
5574 022566 104000                                     HLT
5575 022570 012700 170357         MOV     #170357,R0
5576 022574 052700 170000         BIS     #170000,R0      ;SET BITS 15-12 IF MEM MGMT
5577 022600 012702 000001         10$:   MOV     #1,R2          ;R2 = TEST BIT
5578 022604 030200 1$:           BIT     R2,R0          ;CHECK IF BIT CAN BE SET/CLEARED
5579 022606 001423                                     BEQ    2$
5580 022610 005037 000016         CLR     #TBITVEC+2
5581 022614 030227 000020         BIT     R2,#20        ;CHECK IF TEST WILL SET 'T' BIT
5582 022620 001403                                     BEQ    20$
5583 022622 012737 000002 000016         20$:   MOV     #RTI,#TBITVEC+2 ;SET RTI INTO RETURN
5584 022630 005014                                     CLR     (R4)           ;CLEAR PSW
5585 022632 050214                                     BIS     R2,(R4)        ;SET R2 INTO PSW
5586 022634 011403                                     MOV     (R4),R3        ;GET BIT
5587 022636 020203                                     CMP     R2,R3          ;CHECK THAT BIT WAS SET IN PSW
5588 022640 001401                                     BEQ    .+4
5589 022642 104000                                     HLT
5590 022644 000244                                     CLZ
5591 022646 040214                                     BIC     R2,(R4)        ;CLEAR BIT IN PSW
5592 022650 011403                                     MOV     (R4),R3        ;GET PSW RESULT
5593 022652 001401                                     BEQ    2$              ;BRANCH IF BIC ABOVE CLEARED BIT IN PSW
5594 022654 104000                                     HLT
5595 022656 020227 004000         2$:   CMP     R2,#4000      ;READY TO TEST BIT 12 YET?
5596 022662 001003                                     BNE    12$            ;BRANCH IF NO
5597 022664 012702 030000         MOV     #30000,R2     ;IF YES, SET BITS 12 & 13 TOGETHER
5598 022670 000745                                     BR     1$             ;GO TEST BITS 12 & 13 OF THE PSW
5599 022672 020227 030000         12$:   CMP     R2,#30000   ;READY TO TEST BIT 14?
5600 022676 001003                                     BNE    13$            ;BRANCH IF NO
5601 022700 012702 140000         MOV     #140000,R2   ;IF YES, SET BITS 14 & 15 TOGETHER
5602 022704 000737                                     BR     1$             ;GO TEST BITS 14 & 15 OF THE PSW
5603 022706 006302         13$:   ASL     R2           ;SHIFT TEST BIT
5604 022710 022702 100000         CMP     #100000,R2   ;BRANCH IF ALL BITS NOT TESTED
5605 022714 001333                                     BNE    1$
  
```

# H10

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 124  
 DQKDC.A.P11 07-FEB-77 09:58 T43 CHECK THAT ALL BITS IN THE PSW CAN BE SET AND CLEARED

```

5606 022716 005014          CLR      (R4)          ;CLEAR STATUS
5607 022720 012637 000016  MOV      (SP)+,2@TBITVEC+2 ;RESTORE T BIT RETURN
5608 022724 012746          MOV      (PC)+,-(SP)    ;PUSH ORIGINAL STATUS ON STACK
5609 022726 000000          .WORD   0              ;CONTAINS ORIGINAL PSW
5610 022730 010746          MOV      PC,-(SP)      ;SET RETURN PC
5611 022732 062716 000006  ADD      #6,(SP)
5612 022736 000002          RTI
5613 022740 013704 177776 4$: MOV      2@PSW,R4      ;RETURN
5614 022744 112737 000340 177776 MOVVB   #340,2@PSW     ;SAVE PSW IN R4
5615 022752 004737 054112 JSR      PC,2@CLRTBIT  ;SET PRIORITY LEVEL 6
5616                                     ;GO CLEAR 'T' BIT IF SET
5617                                     ;*****
5618                                     ;*TEST 44 CHECK THAT ALL BITS IN THE CURRENT STACK PTR CAN BE SET CLEARED
5619                                     ;*****
5619 022756                                     TST44:
5620 022756 000004          SCOPE
5621 022760 112737 000044 001202 MOVVB   #44,2@STSTNM
5622 022766 010603          CHKSP: MOV      SP,R3      ;SAVE STACK PTR
5623 022770 000257          CCC
5624 022772 112706 000377          MOVVB   #377,SP      ;SET STACK PTR = -1
5625 022776 006006          1$: ROR      SP          ;ROTATE 0 BIT THROUGH ALL BIT
5626 023000 103776          BCS     1$           ;BIT POSITIONS
5627 023002 005206          INC     SP          ;SHOULD INCREMENT SP TO 0
5628 023004 001403          BEQ     2$
5629 023006 010602          MOV     SP,R2      ;SAVE ERROR STACK PTR
5630 023010 010306          MOV     R3,SP     ;SET STACK PTR FOR TRAP
5631 023012 104000          HLT
5632                                     ;ERROR!
5633 023014 010306          2$: MOV      R3,SP      ;RESTORE ORIGINAL STACK PTR
5634                                     ;CHECK BYTE OPERATIONS USING THE STACK
5635 SPCHK:
5636 023016 010600          MOV     SP,R0      ;SAVE STACK PTR
5637 023020 010003          MOV     R0,R3
5638
5639 023022 005043          CLR     -(R3)
5640 023024 112746 177777          MOVVB   #-1,-(SP)    ;(SP) = 377
5641 023030 022713 000377          CMP     #377,(R3)   ;CHECK THAT ONLY EVEN BYTE WAS AFFECTED
5642 023034 001002          BNE     1$
5643 023036 020306          CMP     R3,SP      ;CHECK AUTO-DEC
5644 023040 001401          BEQ     .+4
5645 023042 104000          1$: HLT
5646
5647 023044 105226          INCB   (SP)+
5648 023046 005723          TST    (R3)+      ;CHECK RESULT
5649 023050 001002          BNE     2$
5650 023052 020006          CMP     R0,SP      ;CHECK AUTO-INC
5651 023054 001401          BEQ     .+4
5652 023056 104000          2$: HLT
5653
5654 023060 005143          COM    -(R3)      ;(R3)=177777
5655 023062 144613          BICB   -(SP),(R3)
5656 023064 022713 177400          CMP     #177400,(R3) ;CHECK RESULT
5657 023070 001002          BNE     3$
5658 023072 020603          CMP     SP,R3
5659 023074 001401          BEQ     .+4
5660 023076 104000          3$: HLT
5661
  
```



```

5662 023100 132627 000377      BITB      (SP)+, #377
5663 023104 001002      BNE       4$
5664 023106 020600      CMP       SP, R0
5665 023110 001401      BEQ       .+4
5666 023112 104000      4$:      HLT
5667
5668 023114 012746 000001      MOV       #1, -(SP)
5669 023120 062706 000002      ADD       #2, SP
5670 023124 012702 177401      MOV       #177401, R2
5671 023130 120246      CMPB      R2, -(SP)
5672 023132 001004      BNE       5$
5673 023134 122602      CMPB      (SP)+, R2
5674 023136 001002      BNE       5$
5675 023140 020006      CMP       R0, SP
5676 023142 001401      BEQ       .+4
5677 023144 104000      5$:      HLT
5678 023146 105037 177776      CLRB      @#PSW
5679 023152 010446      MOV       R4, -(SP)      ;RESTORE ORIGINAL PSW TO STACK
5680 023154 010746      MOV       PC, -(SP)
5681 023156 062716 000006      ADD       #6, (SP)
5682 023162 000002      RTI
5683
5684      ;*****
5685      ;*TEST 45      CHECK THAT 'C' BIT SETS/CLEARs PROPERLY
5686      ;*****
5687      ;ST45:
5688 023164 000004      SCOPE
5689 023166 112737 000045 001202      MOVB      #45, @#STSTNM
5690 023174 012727 177776      CBIT:     MOV       #177776, (PC)+      ;LOAD CONSTANT
5691 023200 000000      1$:      .WORD      0
5692 023202 010700      MOV       PC, R0      ;GET CURRENT PC
5693 023204 162700 000004      SUB       #4, R0      ;POINT R0 TO 1$ ABOVE
5694 023210 005520      2$:      ADC       (R0)+      ;ADD 'C' BIT TO 1$ ABOVE
5695 023212 006340      ASL      -(R0)      ;SHIFT 1$
5696 023214 102375      BVC       2$      ;UNTIL 'V' BIT SETS
5697 023216 022767 077776 177754      CMP       #077776, 1$      ;CHECK RESULT
5698 023224 001401      BEQ       .+4
5699 023226 104000      HLT      ;ERROR! INCORRECT RESULT IN 1$ ABOVE
5700      ;R0=ADDRESS OF DATA
5701
5702      ;CHECK THAT CONDITION CODES ARE SET PROPERLY WHEN A NUMBER (CURRENT PC)
5703      ;AND THAT NUMBER +1 ARE COMPARED, AND VICE VERSA.
5704 023230 010700      CMPN:     MOV       PC, R0      ;GET CURRENT PC
5705 023232 010002      MOV       R0, R2      ;SAVE IN R2
5706 023234 005202      INC       R2      ;MAKE R2 = R0+1
5707 023236 000277      SCC
5708 023240 000251      +CLC!CLN      ;CLEAR C & N BITS
5709 023242 020002      CMP       R0, R2      ;COMPARE # WITH #+1
5710 023244 103003      BCC       1$      ;CARRY BIT SHOULD SET
5711 023246 102402      BVS       1$      ;V BIT SHOULD CLEAR
5712 023250 001401      BEQ       1$      ;Z BIT SHOULD CLEAR
5713 023252 100401      BMI      .+4      ;N BIT SHOULD SET
5714 023254 104000      1$:      HLT      ;ERROR! COMPARE # WITH #+1 FAILED TO
5715      ;SET CONDITION CODES IN PSW CORRECTLY
5716 023256 000277      SCC
5717 023260 120200      CMPB      R2, R0      ;SET CONDITION CODES IN PSW
                        ;COMPARE #+1 WITH #

```



```

5718 023262 103403          BCS      25          ;C BIT SHOULD CLEAR
5719 023264 102402          BVS      25          ;V BIT SHOULD CLEAR
5720 023266 001401          BEQ      25          ;Z BIT SHOULD CLEAR
5721 023270 100001          BPL      .+4        ;N BIT SHOULD CLEAR
5722 023272 104000          25:     HLT          ;ERROR! COMPARE #+1 WITH # FAILED TO SET
5723                                     ;CONDITION CODES IN PSW CORRECTLY
5724 023274 105037 177776          CLRB     @PSW       ;ENSURE PRIORITY 0
5725 023300 000004          RELE4:  SCOPE
5726 023302 010702          MOV      PC,R2
5727 023304 062702 000012          ADD      @12,R2
5728 023310 012707 036574          MOV      @RELOC,PC ;GO RELOCATE PROGRAM CODE
5729 023314 000000          REL44:  .WORD     0
5730                                     ;44444444444444 LAST ADDRESS OF CODE TO BE RELOCATED 444444444444
5731
5732                                     ;*****
5733                                     ;*TEST 46 CHECK EXTENDED INSTRUCTION SET
5734                                     ;*****
5735 023316          TST46:
5736 023316 012767 000001 156000          MOV      @1,STIMES ;;DO 1 ITERATION
5737 023324 000004          SCOPE
5738 023326 112737 000046 001202          MOVB     @46,@STSTNM
5739
5740                                     .SBTTL  START OF SECTION 5
5741          :5555555555555555 FIRST ADDRESS TO BE RELOCATED 5555555555
5742 023334 112737 000046 001202 REL5:  MOVB     @STN-1,@STSTNM
5743 023342 010700          MOV      PC,RO      ;GET PC
5744 023344 005740          TST      -(RO)      ;RO CONTAINS THE ADDRESS OF REL5
5745 023346 010037 001554          MOV      RO,@FRSTAD ;SAVE
5746 023352 010700          MOV      PC,RO      ;GET CURRENT PC
5747 023354 162700 023354          SUB      @,RO        ;SUBTRACT RELOCATION FACTOR
5748 023360 010037 001550          MOV      RO,@FACTOR ;SAVE RELOCATION FACTOR
5749 023364 010737 001212          MOV      PC,@SLPERR ;SET LOOP ADDRESS
5750 023370 062737 000026 001212          ADD      @26,@SLPERR ;ADJUST
5751 023376 013737 001212 001210          MOV      @SLPERR,@SLPADR
5752 023404 105737 001544          TSTB     @NEXEC     ;BR IF TEST CODE TO BE EXECUTED
5753 023410 001402          BEQ      .+6
5754 023412 000167 001366          JMP      RELES
5755 023416 005000          EXTINST: CLR      RO
5756 023420 000277          SCC          ;PRESET CC'S
5757 023422 006700          SXT      RO        ;EXTEND SIGN (1) INTO RO
5758 023424 103005          BCC          ;CHECK RESULT CC'S
5759 023426 102404          BVS          SXTO
5760 023430 001403          BEQ          SXTO
5761 023432 100002          BPL          SXTO
5762 023434 005200          INC      RO        ;CHECK RESULT
5763 023436 001401          BEQ      .+4
5764 023440 104000          SXTO:  HLT
5765
5766 023442 010700          MOV      PC,RO
5767 023444 010002          MOV      RO,R2
5768 023446 012703 177777          MOV      #-1,R3
5769 023452 005102          COM      R2
5770 023454 000243          +CLV!CLC
5771 023456 074003          XOR      RO,R3     ;CLEAR C AND V BITS
5772 023460 103404          BCS          ;R3 SHOULD CONTAIN COMPLEMENT OF RO
5773 023462 102403          BVS          ;CHECK THAT C WAS NOT AFFECTED
                    ;AND THAT V WAS CLEARED

```

# K10

5774	023464	001402		BEQ	XOR0	
5775	023466	020203		CMP	R2,R3	;CHECK RESULT
5776	023470	001401		BEQ	.+4	
5777	023472	104000		XOR0:	HLT	;ERROR! XOR FAILED
5778						
5779	023474	010700		MOV	PC,R0	
5780	023476	022020		CMP	(R0)+,(R0)+	;SET ADDRESS REGISTER
5781	023500	000401		BR	15	;RESERVE WORD FOR TEST DATA
5782	023502	000000		.WORD	0	;CONTAINS TEST DATA
5783	023504	005700		15:	TST	R0
5784	023506	006710		SXT	(R0)	;EXTEND SIGN OF ADDRESS INTO
5785	023510	005002		CLR	R2	;ADDRESS (R0)=-1 IF MSB R0=1
5786	023512	005700		TST	R0	;OTHERWISE, (R0)=0
5787	023514	100001		BPL	.+4	;CHECK SIGN OF ADDRESS
5788	023516	005102		COM	R2	;COMPLEMENT CHECK REG IF NEG
5789	023520	021002		CMP	(R0),R2	;CHECK RESULT OF SXT
5790	023522	001401		BEQ	.+4	
5791	023524	104000		SXT1:	HLT	;ERROR! SXT FAILED TO EXTEND SIGN PROPERLY
5792						
5793	023526	012710	100000	MOV	#100000,(R0)	;PRESET DATA
5794	023532	011002		MOV	(R0),R2	
5795	023534	000277		SCC		;PRESET CC'S
5796	023536	074210		XOR	R2,(R0)	;XOR 100000 WITH 100000 RESULT = 0
5797	023540	103007		BCC	XOR1	;CHECK CC'S AFTER XOR
5798	023542	102406		BVS	XOR1	
5799	023544	001005		BNE	XOR1	
5800	023546	100404		BMI	XOR1	
5801	023550	005710		TST	(R0)	;CHECK RESULT (0)
5802	023552	001002		BNE	XOR1	
5803	023554	005402		NEG	R2	;CHECK THAT REG WAS NOT AFFECTED
5804	023556	102401		BVS	.+4	
5805	023560	104000		XOR1:	HLT	
5806						
5807	023562	010702		MOV	PC,R2	
5808	023564	022222		CMP	(R2)+,(R2)+	
5809	023566	000401		BR	SXT4	;PRESERVE WORD FOR DATA
5810	023570	000000		.WORD	0	;RESERVED FOR DATA
5811	023572	012722	125252	SXT4:	MOV	#125252,(R2)+
5812	023576	006742		SXT	-(R2)	;PRESET DATA
5813	023600	074722		XOR	PC,(R2)+	;EXTEND SIGN
5814	023602	010700		MOV	PC,R0	;GET PC
5815	023604	005740		TST	-(R0)	;SUBTRACT 2 FROM PC
5816	023606	005100		COM	R0	;R0=RESULT OF XOR PC-1 ABOVE
5817	023610	074042		XOR	R0,-(R2)	;CHECK RESULT OF SXT AND XOR ABOVE
5818	023612	001401		BEQ	.+4	
5819	023614	104000		XOR24:	HLT	;ERROR! SXT & XOR ABOVE INCORRECT
5820						
5821	023616	012704	000001	MOV	#1,R4	;SET R4
5822	023622	006767	000060	SXT	XOR6A	;PRESET DATA=0
5823	023626	074467	000054	XOR	R4,XOR6A	
5824	023632	100423		BMI	XOR6	
5825	023634	006304		ASL	R4	;SHIFT R4
5826	023636	102373		BVC	25	;UNTIL V SETS (R4=100000)
5827	023640	100020		BPL	XOR6	;BRANCH IF 'N' IS CLEAR
5828	023642	074467	000040	XOR	R4,XOR6A	;XOR6A=177777
5829	023646	100015		BPL	XOR6	



5830	023650	074767	000032		XOR	PC, XOR6A	; XOR PC WITH XOR6A (177777)
5831	023654	010767	000030		MOV	PC, XOR6B	; FORM PC AS USED IN XOR ABOVE
5832	023660	162767	000004	000022	SUB	#4, XOR6B	
5833	023666	005167	000016		COM	XOR6B	
5834	023672	026767	000012	000006	CMP	XOR6B, XOR6A	; XOR6A SHOULD = COMPLEMENT OF PC
5835	023700	001401			BEQ	.+4	
5836	023702	104000			XOR6: HLT		; ERROR! XOR TESTS ABOVE FAILED
5837							
5838	023704	000402			BR	.+6	
5839							
5840	023706	000000			XOR6A: .WORD	0	; CONTAINS DATA USED BY TEST ABOVE
5841	023710	000000			XOR6B: .WORD	0	
5842							
5843							
5844	023712	012700	077777		MOV	#077777, R0	; SET SOURCE OPERAND FOR ADD
5845	023716	006767	177764		SXT	XOR6A	; CLEAR XOR6A
5846	023722	001004			BNE	SXT6	; CHECK CC'S AFTER EXTENDING ZERO'S
5847	023724	100403			BMI	SXT6	
5848	023726	103402			BCS	SXT6	
5849	023730	102401			BVS	SXT6	
5850	023732	000401			BR	.+4	
5851	023734	104000			SXT6: HLT		; ERROR! SXT FAILED
5852							
5853	023736	012702	000001		MOV	#1, R2	; SET DEST OPERAND FOR ADD
5854	023742	013703	001550		MOV	#FACTOR, R3	; LOAD INDEX REGISTER
5855	023746	060002			ADD	R0, R2	; RESULT OF ADD=100000
5856	023750	006763	023706		SXT	XOR6A(3)	; EXTEND SIGN OF ADD ABOVE
5857	023754	001403			BEQ	SXT6A	
5858	023756	005267	177724		INC	XOR6A	; CHECK RESULT OF SXT
5859	023762	001401			BEQ	.+4	
5860	023764	104000			SXT6A: HLT		; ERROR! SXT ABOVE FAILED TO EXTEND ; SIGN
5861							
5862	023766	010703			MOV	PC, R3	
5863	023770	000402			BR	.+6	; PRESERVE 2 WORDS FOR DATA
5864	023772	000000			SXRA: .WORD	0	; RESERVED WORD FOR DATA
5865	023774	000000			SXRB: .WORD	0	; RESERVED WORD FOR DATA
5866	023776	005723			TST	(R3)+	
5867	024000	010304			MOV	R3, R4	; R3 = ADDRESS OF SXRA
5868	024002	000250			CLN		; CLEAR N BIT
5869	024004	006724			SXT	(R4)+	; EXTEND ZEROS INTO SXRA
5870	024006	001401			BEQ	.+4	
5871	024010	104000			SXT2: HLT		; ERROR! SXT FAILED
5872							
5873	024012	010467	177754		MOV	R4, SXRA	; SXRA = ADDRESS OF SXRB
5874	024016	000257			CCC		; CLEAR CONDITION CODES
5875	024020	006733			SXT	2(R3)+	; EXTEND ZEROS INTO SXRB
5876	024022	001401			BEQ	.+4	
5877	024024	104000			SXT3: HLT		; ERROR!
5878							
5879	024026	000270			SEN		; SET N BIT
5880	024030	006753			SXT	2-(R3)	; EXTEND ONES INTO SXRB
5881	024032	100401			BMI	.+4	
5882	024034	104000			SXT5: HLT		; ERROR!
5883							
5884	024036	012704	025252		MOV	#025252, R4	; R4 = 025252
5885	024042	074433			XOR	R4, 2(R3)+	; SXRB = 152525 (COMPLEMENT OF R4)



```

5886 024044 005002 CLR R2
5887 024046 074253 XOR R2,2-(R3) ;SXR B REMAINS UNCHANGED
5888 024050 001405 BEQ XOR35 ;CHECK CONDITION CODES
5889 024052 100004 BPL XOR35
5890 024054 005104 COM R4 ;R4 = 152525
5891 024056 020467 177712 CMP R4,SXR B ;CHECK XOR
5892 024062 001401 BEQ .+4
5893 024064 104000 XOR35: HLT ;ERROR! XOR FAILED
5894
5895 024066 005743 TST -(R3) ;R3 = ADDRESS OF SXRA-2
5896 024070 000250 CLN ;CLEAR N BIT
5897 024072 006773 000002 SXT 22(R3) ;SXR B = 0
5898 024076 001401 BEQ .+4
5899 024100 104000 SXT7: HLT ;ERROR! SXT FAILED
5900
5901 024102 074473 000002 XOR R4,22(R3) ;SXR B = R4
5902 024106 020473 000002 CMP R4,22(R3) ;CHECK XOR
5903 024112 001401 BEQ .+4
5904 024114 104000 XOR7: HLT ;ERROR! XOR FAILED
5905
5906 *****
5907 *TEST 47 SOB TEST
5908 * NOTE: DO NOT INSERT ANY CODE IN FOLLOWING SOB TESTS
5909 * SINCE IT TESTS THE MAXIMUM BRANCH WIDTH OF THE INSTRUCTION.
5910 *****
5911 TST47:
5912 024116 000004 SCOPE
5913 024120 112737 000047 001202 MOVB #47,2#STSTNM
5914 024126 005005 CLR R5 ;CLEAR ERROR INDICATOR
5915 024130 000407 BR SOB0 ;BRANCH TO SOB TEST
5916
5917 024132 005004 SOB10: CLR R4 ;R4 = 0
5918 024134 005705 TST R5 ;CHECK ERROR INDICATOR
5919 024136 001401 BEQ .+4 ;SOB BRANCHED CORRECTLY
5920 024140 104000 HLT ;ERROR!
5921
5922 024142 005005 SOB9: CLR R5 ;CLEAR INDICATOR (R5)
5923 024144 006004 ROR R4 ;ROTATE RIGHT R4
5924 024146 000467 BR SOB8
5925
5926 024150 012700 000010 SOB0: MOV #10,R0 ;R0=10
5927 024154 000277 SCC ;SET CONDITION CODES
5928 024156 001012 SOB1: BNE SOB2 ;CHECK CONDITION CODES AFTER SOB
5929 024160 100011 BPL SOB2 ;SOB SHOULD NOT EFFECT THE
5930 024162 102010 BVC SOB2 ;CONDITION CODES.
5931 024164 103007 BCC SOB2
5932 024166 077005 SOB RO,SOB1
5933 024170 001005 BNE SOB2 ;CHECK CONDITION CODES AFTER
5934 024172 100004 BPL SOB2 ;SOB FALLS THROUGH.
5935 024174 102003 BVC SOB2 ;SOB SHOULD NOT EFFECT
5936 024176 103002 BCC SOB2 ;CONDITION CODES.
5937 024200 005700 TST R0 ;CHECK IF R0=0
5938 024202 001401 BEQ .+4
5939 024204 104000 SOB2: HLT ;ERROR!
5940
5941 024206 012702 000100 MOV #100,R2 ;R2=100

```

```

5942 024212 012700 000101          MOV      #101,R0      ;SET CHECK REGISTER, R0=101
5943 024216 001414          SOB3:   BEQ      SOB4      ;CHECK CONDITION CODES AFTER
5944 024220 100413          BMI      SOB4      ;SOB BRANCH,
5945 024222 102412          BVS      SOB4      ;SOB SHOULD NOT EFFECT
5946 024224 103411          BCS      SOB4      ;CONDITION CODES.
5947 024226 005300          DEC      R0          ;DECREMENT CHECK REGISTER
5948 024230 020002          CMP      R0,R2      ;CHECK THAT SOB DECREMENTS
5949 024232 001006          BNE      SOB4
5950 024234 000257          CCC
5951 024236 077211          SOB      R2,SOB3     ;SET CONDITION CODES BEFORE SOB
5952 024240 001403          BEQ      SOB4      ;BRANCH TO SOB3 UNTIL R2=0
5953 024242 100402          BMI      SOB4      ;CHECK CONDITION CODES AFTER
5954 024244 005702          TST      R2          ;SOB FALLS THROUGH
5955 024246 001401          BEQ      .+4        ;CHECK IF R2=0
5956 024250 104000          SOB4:   HLT
5957
5958 024252 012700 000001          SOB5:   MOV      #1,R0      ;R0=1
5959 024256 000401          BR      .+4
5960 024260 104000          HLT
5961 024262 077002          SOB      R0,.-2     ;ERROR!
5962
5963 024264 005700          TST      R0          ;SOB SHOULD NOT BRANCH
5964 024266 001401          BEQ      .+4        ;CHECK IF R0=0 AFTER SOB
5965 024270 104000          HLT
5966
5967 024272 012704 100000          SOB5A:  MOV      #100000,R4 ;R4=100000
5968 024276 000403          BR      1$
5969 024300 005204          3$:    INC      R4
5970 024302 100403          BMI      2$
5971 024304 104000          HLT
5972
5973
5974 024306 077404          1$:    SOB      R4,3$
5975 024310 104000          HLT
5976
5977 024312 012703 000100          2$:    MOV      #100,R3     ;R3=100
5978 024316 077301          SOB6:   SOB      R3,SOB6   ;USE SOB TO BRANCH TO ITSELF
5979 024320 005703          TST      R3          ;CHECK IF R3=0
5980 024322 001703          BEQ      SOB10
5981 024324 104000          SOB7:   HLT
5982
5983 024326 005705          SOB8:   TST      R5
5984
5985
5986
5987
5988
5989 024330 001401          BEQ      .+4        ;CHECK INDICATOR (R5)
5990 024332 104000          HLT
5991
5992 024334 005205          INC      R5          ;IF SOB BRANCHES INCORRECTLY
5993 024336 077477          SOB      R4,SOB9   ;WHEN CHECKING MAX. BRANCH,
5994 024340 005704          TST      R4          ;R5 WILL NOT BE CLEARED AT
5995 024342 001401          BEQ      .+4        ;THIS POINT INDICATING AN ERROR.
5996 024344 104000          HLT
5997

```

;;\*\*\*\*\*



```

5998
5999
6000 024346
6001 024346 000004
6002 024350 112737 000050 001202
6003 024356 010602
6004 024360 010705
6005 024362 010500
6006 024364 010546
6007 024366 010746
6008 024370 010746
6009 024372 010746
6010 024374 010746
6011 024376 010746
6012 024400 012746 006405
6013 024404 010605
6014 024406 004767 000002
6015 024412 000403
6016 024414 000205
6017 024416 104000
6018 024420 000407
6019 024422 020602
6020 024424 001402
6021 024426 104000
6022 024430 000403
6023 024432 020005
6024 024434 001401
6025 024436 104000
6026 024440 010206
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051 024442
6052 024442 000004
6053 024444 112737 000051 001202

```

```

;*TEST 50 CHECK THE MARK INSTRUCTION
*****
TST50:
SCOPE
MOV #50,2#STSTNM
MRKTST: MOV SP,R2
MOV PC,R5 ;THE STACK LOOKS LIKE THIS AFTER
MOV R5,RO ;THE JSR INSTRUCTION
MOV R5,-(SP) ; -2(SP)= R0 THIS IS A
MOV PC,-(SP) ; -4(SP)= PC STRING
MOV PC,-(SP) ; -6(SP)= PC+2 OF
MOV PC,-(SP) ; -10(SP)= PC+4 FIVE
MOV PC,-(SP) ; -12(SP)= PC+6 DUMMY
MOV PC,-(SP) ; -14(SP)= PC+10 ARGUMENTS
MOV #MARK+5,-(SP) ; -16(SP)= MARK 5
MOV SP,R5 ; -20(SP)= PC PUSHED BY JSR
JSR PC,MARK1
BR +10
MARK1: RTS R5 ;ERROR! SHOULD BE DOING MARK 5 INST.
HLT
BR MARKEX
CMP SP,R2
BEQ +6
HLT ;ERROR! SP NOT RETURNED TO PROPER
BR MARKEX ;VALUE BY MARK INSTRUCTION
CMP RO,R5
BEQ +4
HLT ;ERROR! DID NOT RESTORE R5 FROM STACK
MARKEK: MOV R2,SP ;RESTORE SP
*****
;*TEST 51 RTT/RTI TEST
*****
RTT/RTI TEST INSURES THAT CP DOES THE INSTRUCTION FOLLOWING
AN RTT IF THE "T"BIT IS SET IN THE PSW,BUT DOES HONOR
THE TRAP IMMEDIATELY IF IT EXECUTES AN RTI
INSTRUCTION SEQUENCE-RTT
2S: RTT ;NO 'T' TRAP AFTER RTT
INC RO ;RO=000001
; 'T' TRAP TO 5S AFTER INC
5S: COM RO ;RO=177776
MOV SAVPSW,2(SP) ;CLEAR 'T' BIT IN RETURN PSW
RTI ;RETURN TO INSTRUCTION FOLLOWING INC
CMP #RTT,2S ;CHECK
ETC

INSTRUCTION SQUENCE-RTI
2S: RTI ;'T' TRAP AFTER RTI
5S: COM RO ;RO=177777
MOV SAVPSW,2(SP) ;CLEAR 'T' BIT IN RETURN PSW
RTI ;RETURN TO INC INSTRUCTION
INC RO ;RO=000000
CMP #RTT,2S ;CHECK
ETC
*****
TST51:
SCOPE
MOV #51,2#STSTNM

```



```

6054 024452 013767 177776 000174 RTT1: MOV      2#PSW, SAVPSW      ;SAVE PSW
6055 024460 032767 000020 000166 BIT      #20, SAVPSW      ;CHECK IF "T"BIT SET
6056 024466 001143          BNE      RTT2EX          ;BRANCH TO EXIT
6057 024470 010746          1S:  MOV      PC, -(SP)      ;GET CURRENT PC
6058 024472 062716 000116 ADD      #5, -(SP)      ;FORM RELOCATED PC
6059 024476 012637 000014 MOV      (SP)+, 2#TBITVEC ;LOAD INTO TRAP VECTOR
6060 024502 016746 000146 MOV      SAVPSW, -(SP)   ;GET CURRENT PSW
6061 024506 011637 000016 MOV      (SP), 2#TBITVEC+2
6062 024512 052737 000340 177776 BIS      #PR7, 2#PSW     ;SET PRIORITY LEVEL 7
6063 024520 005000          CLR      RO
6064 024522 052716 000360 BIS      #PR7+20, (SP)   ;SET "T"BIT IN PSW ON STACK
6065 024526 010746          MOV      PC, -(SP)      ;PUT THE PC ON THE STACK
6066 024530 062716 000006 ADD      #6, (SP)      ;ADJUST PC FOR NEXT INSTRUCTION
6067 024534 000006          2S:  RTT
6068 024536 005200          INC      RO
6069          ;DONE TO SEE IF INSTR. FOLLOWING
6070 024540 042737 000340 177776 BIC      #PR7, 2#PSW     ;RTT IS EXECUTED IF T-BIT SET
6071 024546 022767 000006 177760 CMP      #RTT, 2S      ;SET PRIORITY LEVEL 0
6072 024554 001005          BNE      3S
6073 024556 022700 177776 CMP      #177776, RO
6074 024562 001406          BEQ      4S
6075 024564 104000          HLT
6076 024566 000415          BR      6S
6077 024570 005700          3S:  TST      RO
6078          ;CHECK IF INC WAS EXECUTED
6079 024572 001413          BEQ      6S            ;CHECK IF COM-RO EXECUTED
6080 024574 104000          HLT
6081 024576 000411          BR      6S
6082 024600 012767 000002 177726 4S:  MOV      #RTI, 2S
6083 024606 000730          BR      1S
6084 024610 005100          5S:  COM      RO
6085 024612 016766 000036 000002 MOV      SAVPSW, 2(SP) ;RTT CHECK
6086 024620 000002          RTI
6087 024622 012767 000006 177704 6S:  MOV      #RTT, 2S
6088 024630 012737 001534 000014 MOV      #SRTN, 2#TBITVEC ;RESTORE 'T' TRAP VECTOR
6089 024636 005037 000016 CLR      2#TBITVEC+2
6090          RTT1EX:
6091          ;*****
6092          ;#TEST 52 SECOND RTT TEST
6093          ;*****
6094          †S2S2:
6095 024642 000004          SCOPE
6096 024644 112737 000052 001202 MOVB     #52, 2#STSTNM
6097 024652 000401          BR      RTT2A
6098 024654 000000          SAVPSW: .WORD 0
6099 024656 016700 177772 RTT2A:  MOV      SAVPSW, RO ;GET SAVED PSW
6100 024662 105000          CLRB     RO ;CLEAR PRIORITY LEVEL, T, AND COND CODES
6101 024664 012702 140000          MOV      #UM, R2

```

D11

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 133  
DQKDC.A.P11 07-FEB-77 09:58 T52 SECOND RTT TEST

6102 024670 074002  
6103 024672 001423  
6104 024674 032700 140000  
6105 024700 001036

XOR R0,R2  
BEQ 2\$  
BIT #UM,R0  
BNE RTT2EX

;USER MODE



```

6106
6107 ;TEST THAT RTT CLEARS BITS 12,13 & PRIORITY LEVEL BITS IN KERNEL MODE
6108 024702 012737 030240 177776 MOV #PUM+PR5,2#PSW ;GO TO KERNEL MODE
6109 024710 012746 000100 MOV #PR2, -(SP)
6110 024714 010746 MOV PC, -(SP)
6111 024716 062716 000006 ADD #1$, -(SP) ;FORM NEW PC
6112 024722 000006 RTT
6113 024724 013700 177776 1$: MOV 2#PSW, R0 ;NOW USING REG SET 0
6114 024730 022700 000100 CMP #PR2, R0 ;TESTS THE PSW AFTER THE RTT
6115 024734 001420 BEQ RTT2EX
6116 024736 104000 HLT ;ERROR! INCORRECT PSW AFTER THE RTT
6117 024740 000416 BR RTT2EX
6118
6119 ;TEST TO INSURE THAT RTI DOES NOT CLEAR BITS 12-15 IN USER MODE
6120 024742 052737 030340 177776 2$: BIS #PUM+PR7,2#PSW
6121 024750 005046 CLR -(SP)
6122 024752 010746 MOV PC, -(SP)
6123 024754 062716 000006 ADD #5$, -(SP)
6124 024760 000002 RTI ;ATTEMPS TO INSERT A PSW OF 0
6125 024762 022737 170340 177776 5$: CMP #UM+PUM+PR7,2#PSW
6126 024770 001402 BEQ RTT2EX
6127 024772 104000 HLT ;ERROR! RTI CLEARED BITS IN PSW
6128 024774 000400 BR RTT2EX
6129
6130 024776 016737 177652 177776 RTT2EX: MOV SAVPSW,2#PSW
6131 025004 000004 RELE5: SCOPE
6132 025006 010702 MOV PC, R2
6133 025010 062702 000012 ADD #12, R2
6134 025014 012707 036574 MOV #RELOC, PC ;GO RELOCATE PROGRAM CODE
6135 025020 000000 REL55: .WORD 0
6136 ;55555555555555 LAST ADDRESS OF CODE TO BE RELOCATED 5555555555
6137
6138 ;*****
6139 ;*TEST 53 CHECK ASH, ASHC, MUL, AND DIV INSTRUCTIONS
6140 ;*****
6141 025022 †ST53:
6142 025022 012767 000001 154274 MOV #1, $TIMES ;;DO 1 ITERATION
6143 025030 000004 SCOPE
6144 025032 112737 000053 001202 MOVB #53,2#$STSTNM
6145
6146 .SBTTL START OF SECTION 6
6147 :6666666666666666 FIRST ADDRESS TO BE RELOCATED 6666666666
6148 025040 REL6: MOVB #STN-1,2#$STSTNM
6149 025046 010700 MOV PC, R0 ;GET PC
6150 025050 005740 TST -(R0) ;R0 CONTAINS THE ADDRESS OF REL6
6151 025052 010037 001554 MOV R0,2#FRSTAD ;SAVE
6152 025056 010700 MOV PC, R0 ;GET CURRENT PC
6153 025060 162700 025060 SUB #, R0 ;SUBTRACT RELOCATION FACTOR
6154 025064 010037 001550 MOV R0,2#FACTOR ;SAVE RELOCATION FACTOR
6155 025070 010737 001212 MOV PC,2#$LPERR ;SET LOOP ADDRESS
6156 025074 062737 000026 001212 ADD #26,2#$LPERR ;ADJUST
6157 025102 013737 001212 001210 MOV 2#$LPERR,2#$LPADR
6158 025110 105737 001544 TSTB 2#NEXEC ;BR IF TEST CODE TO BE EXECUTED
6159 025114 001402 BEQ .+6
6160 025116 000167 004402 JMP RELE6
6161 025122 012700 000001 ASHLO: MOV #1, R0 ;R0 WILL BE THE SHIFT COUNT

```





```

6218 025354 073400
6219 025356 006303
6220 025360 006102
6221 025362 077003
6222 025364 020402
6223 025366 001002
6224 025370 020503
6225 025372 001401
6226 025374 104000
6227 025376 005216
6228 025400 021666 000002
6229 025404 001356
6230 025406 022626
6231
6232 025410 012746 177740
6233 025414 012746 177777
6234 025420 011600
6235 025422 010702
6236 025424 010204
6237 025426 005003
6238 025430 005005
6239 025432 000262
6240 025434 073200
6241 025436 102410
6242 025440 005400
6243 025442 006204
6244 025444 006005
6245 025446 077003
6246 025450 020204
6247 025452 001002
6248 025454 020305
6249 025456 001401
6250 025460 104000
6251 025462 005316
6252 025464 021666 000002
6253 025470 001353
6254 025472 022626
6255
6256
6257
6258
6259
6260
6261 025474
6262 025474 000004
6263 025476 112737 000054 001202
6264 025504 012700 000001
6265 025510 012706 000700
6266 025514 005016
6267 025516 010702
6268 025520 010227
6269 025522 000000
6270 025524 005003
6271 025526 005004
6272 025530 010205
6273 025532 100001

2S: ASHC R0,R4 ;SHIFT R4 LEFT AS SPECIFIED BY R0
ASL R3 ;SHIFT R2,R3 LEFT
ROL R2 ;AS SPECIFIED BY R0
SOB R0,2S
CMP R4,R2 ;CHECK RESULTS
BNE 3S
CMP R5,R3
BEQ .+4

3S: HLT
INC (SP) ;INCREMENT NEXT PASS SHIFT COUNT
CMP (SP),2(SP) ;REACHED MAX COUNT (31.)
BNE 1S
CMP (SP)+,(SP)+ ;RESTORE STACK PTR

ASHCRD: MOV #-32,-(SP) ;PUT MAX RIGHT SHIFT COUNT ON STACK
MOV #-1,-(SP) ;PUT PASS SHIFT COUNT ON STACK
1S: MOV (SP),R0 ;GET PASS SHIFT COUNT
MOV PC,R2 ;R2,R3 & R4,R5 ARE THE DATA REGISTERS
MOV R2,R4 ;TO BE SHIFTED BY TEST
CLR R3
CLR R5
SEV ;SET V BIT IN PSW
ASHC R0,R2 ;SHIFT R2,R3 RIGHT R0 TIMES
BVS 3S ;SHIFT RIGHT CLEARS V
NEG R0 ;NEGATE SHIFT COUNT FOR SOB
2S: ASR R4 ;SHIFT R4,R5 RIGHT R0 TIMES
ROR R5
SOB R0,2S
CMP R2,R4 ;CHECK RESULT
BNE 3S
CMP R3,R5
BEQ .+4

3S: HLT
DEC (SP) ;SET SHIFT COUNT FOR NEXT PASS
CMP (SP),2(SP) ;CHECK IF MAX SHIFT COUNT
BNE 1S
CMP (SP)+,(SP)+ ;RESTORE STACK PTR
*****
*TEST 54 CHECK MUL
* THE BELOW TEST OF THE MUL INSTRUCTION MULTIPLIES THE CURRENT PC
* BY 1,2,4,8 ETC AND SHIFTS THE SAME PC VALUE USING AN ASHC LEFT BY
* 0,1,2,3,ETC AND COMPARES THE RESULTS. CONDITION CODE RESULTS ARE NOT CHECKED.
*****
TST54: SCOPE
MOV #54,#STSTNM ;R0 CONTAINS MULTIPLIER FOR MUL
MULO: MOV #1,R0 ;SETUP THE STACK
MOV #USESTK,SP ;(SP) CONTAINS SHIFT VALUE FOR ASHC
CLR (SP) ;R3,R2 & R5,R4 ARE DATA REGISTERS
1S: MOV PC,R2 ;SAVE MULTIPLICAND
MOV R2,(PC)+ ;CONTAINS ORIGINAL MULTIPLICAND
WORD 0
CLR R3
CLR R4
MOV R2,R5 ;FOR MUL AND ASHC
BPL .+4 ;IF MULTIPICAND IS NEG THEN SET R4 = -1

```



```

6274 025534 005104          COM      R4          ;FOR ASHC
6275 025536 000277          SCC          ;PRESET CC'S
6276 025540 070200          MUL      R0,R2      ;MULTIPLY R2 BY R0 LEAVE PRODUCT
6277                                ;IN R2,R3 MSH IN R2,LSH IN R3
6278 025542 102406          BVS      2$          ;
6279 025544 001405          BEQ      2$          ;PRODUCT WILL NEVER BE = 0
6280 025546 073416          ASHC     (SP),R4     ;'MULTIPLY' R4,R5 BY (SP) LEAVE PRODUCT
6281                                ;IN R4,R5 MSH IN R4,LSH IN R5
6282 025550 020204          CMP      R2,R4      ;CHECK MSH RESULT
6283 025552 001002          BNE      2$          ;
6284 025554 020305          CMP      R3,R5      ;CHECK LSH RESULT
6285 025556 001401          BEQ      .+4        ;
6286 025560 104000          HLT          ;
6287 025562 005216          INC      (SP)       ;INCREMENT ASHC SHIFT COUNT
6288 025564 006300          ASL      R0          ;SHIFT MUL MULTIPLIER
6289 025566 102353          BVC      1$          ;
6290                                ;CHECK MUL INST WITH MULTIPLIER (R0) = 100000
6291 025570 010702          MOV      PC,R2      ;R2 = MULTIPLICAND
6292 025572 005202          INC      R2          ;
6293 025574 010227          MOV      R2,(PC)+   ;SAVE MULTIPLICAND
6294 025576 000000          .WORD    0          ;CONTAINS ORIGINAL MULTIPLICAND
6295 025600 005103          COM      R3          ;
6296 025602 010204          MOV      R2,R4      ;R4 WILL BE MSH 'PRODUCT'
6297 025604 006204          ASR      R4          ;FORM 'PRODUCT'
6298 025606 005104          COM      R4          ;COMPLEMENT MSH 'PRODUCT'
6299 025610 070200          MUL      R0,R2      ;MULTIPLY R2 BY 100000 LEAVING
6300                                ;R2 = MSH, R3 = LSH PRODUCT
6301 025612 020204          CMP      R2,R4      ;COMPARE MSH PRODUCTS
6302 025614 001002          BNE      3$          ;
6303 025616 020003          CMP      R0,R3      ;CHECK LSH PRODUCT
6304 025620 001401          BEQ      .+4        ;
6305 025622 104000          HLT          ;
6306                                ;*****
6307                                ;*TEST 55 CHECK THE DIV INSTRUCTION
6308                                ;*
6309                                ;* THE BELOW TEST OF THE DIV INSTRUCTION DIVIDES THE CURRENT PC BY
6310                                ;* 1,2,4,8 ETC LEAVING THE QUOTIENT/REMAINDER IN R2/R3. NEXT THE QUOTIENT
6311                                ;* IS MULTIPLIED BY 1,2,4,8 ETC AND THE REMAINDER ADDED. THE RESULT IS
6312                                ;* THEN COMPARED WITH THE ORIGINAL CURRENT PC.
6313                                ;*****
6314 025624 000004          TST55:  SCOPE
6315 025626 112737 000055 001202  MOV      #55,2#STSTNM
6316 025634 012700 000001          DIV0:  MOV      #1,R0          ;R0=DIVISOR
6317 025640 010716          MOV      PC,(SP)     ;SAVE DATA ON STACK
6318 025642 011603          1$:    MOV      (SP),R3    ;GET DATA
6319 025644 005002          CLR      R2          ;CLEAR MSH DIVIDEND
6320 025646 000277          SCC          ;
6321 025650 071200          DIV      R0,R2      ;DIVIDE R2 BY R0 LEAVING QUOTIENT IN R2
6322                                ;AND REMAINDER IN R3
6323 025652 103417          BCS      2$          ;
6324 025654 100416          BMI      2$          ;
6325 025656 102007          BVC      20$         ;BRANCH IF DIVIDE WORKED
6326 025660 022700 000001          CMP      #1,R0      ;V BIT SHOULD ONLY SET IF DIVIDING BY 1
6327 025664 001012          BNE      2$          ;AND THE LSH OF DIVIDEND
6328 025666 032716 100000          BIT      #100000,(SP);IS NEGATIVE
6329 025672 001407          BEQ      2$          ;

```

6330	025674	000407			BR	3\$			
6331	025676	010204			20\$: MOV	R2,R4			;GET QUOTIENT
6332	025700	070400			MUL	R0,R4			;MULTIPLY QUOTIENT BY DIVISOR
6333	025702	060305			ADD	R3,R5			;ADD REMAINDER TO LSH PRODUCT
6334	025704	103402			BCS	2\$			;SHOULD BE NO CARRY
6335	025706	021605			CMP	(SP),R5			;CHECK RESULT
6336	025710	001401			BEQ	+.4			
6337	025712	104000			2\$: HLT				;ERROR! DIVIDE FAILED
6338									;QUOTIENT IS IN R2,REMAINDER IN R3
6339									;ORIGINAL PC IS ON STACK AND FINAL
6340									;PRODUCT IN R4,R5 [MSH][LSH]
6341	025714	006300			3\$: ASL	R0			;GET NEXT DIVISOR
6342	025716	102351			BVC	1\$			
6343									
6344									
6345	025720	005016			:CHECK ASH,ASHC,MUL AND DIV INSTRUCTIONS USING ADDRESS MODE 1				
6346	025722	005000			ASHL1: CLR	(SP)			; (SP) = SHIFT COUNT
6347	025724	012702	000020		CLR	R0			;R0 = SHIFT COUNT FOR CHECK ASH
6348	025730	005067	000012		1\$: MOV	#16.,R2			;R2 = MAX LEFT SHIFT COUNT
6349	025734	010703			CLR	2\$			;CLEAR CC'S HOLDING ADDRESS
6350	025736	010304			MOV	PC,R3			;R3,R4 = DATA TO BE SHIFTED
6351	025740	072316			MOV	R3,R4			
6352	025742	013727	177776		ASH	(SP),R3			;SHIFT R3 LEFT (SP) TIMES
6353	025746	000000			2\$: MOV	#PSW,(PC)+			;SAVE CC'S
6354					.WORD	0			;CONTAINS ASH (SP),R3 CC'S IN EVEN BYTE
6355	025750	072400							;AND ASH R0,R4 CC'S IN ODD BYTE
6356	025752	113767	177776	177767	ASH	R0,R4			;SHIFT R4 LEFT R0 TIMES
6357	025760	020304			MOV	#PSW,2\$+1			;SAVE CC'S IN ODD BYTE OF 2\$
6358	025762	001004			CMP	R3,R4			;COMPARE RESULTS
6359	025764	126767	177756	177755	BNE	3\$			;BRANCH IF THEY DO NOT COMPARE
6360	025772	001401			CMP	2\$,2\$+1			;CHECK CC'S AFTER ASH INSTRUCTIONS
6361	025774	104000			3\$: BEQ	+.4			
6362					HLT				;ERROR! EITHER RESULTS OF SHIFT OR
6363	025776	005200							;RESULT CC'S ARE INCORRECT
6364	026000	005216			INC	R0			;INCREMENT SHIFT COUNT FOR ASH R0,R4
6365	026002	020200			INC	(SP)			;INCREMENT SHIFT COUNT FOR ASH (SP),R3
6366	026004	001351			CMP	R2,R0			;CHECK FOR MAX SHIFT COUNT
6367					BNE	1\$			
6368	026006	005016			ASHR1: CLR	(SP)			; (SP) = SHIFT COUNT FOR ASH (SP),R4
6369	026010	005000			CLR	R0			;R0 = SHIFT COUNT FOR ASH R0,R5
6370	026012	005402			NEG	R2			;R2 = MAX RIGHT SHIFT COUNT (SET BY
6371									;ABOVE TEST TO 16. NOW = -16.
6372	026014	005067	000012		1\$: CLR	2\$			;CLEAR CC'S HOLDING ADDRESS
6373	026020	010704			MOV	PC,R4			;R4,R5 = DATA TO BE SHIFTED RIGHT
6374	026022	010405			MOV	R4,R5			
6375	026024	072416			ASH	(SP),R4			;SHIFT R4 RIGHT (SP) TIMES
6376	026026	013727	177776		MOV	#PSW,(PC)+			;SAVE CC'S
6377	026032	000000			2\$: .WORD	0			;CONTAINS ASH (SP),R4 CC'S IN EVEN BYTE
6378									;AND ASH R0,R5 CC'S IN ODD BYTE
6379	026034	072500			ASH	R0,R5			;SHIFT R5 RIGHT R0 TIMES
6380	026036	113767	177776	177767	MOV	#PSW,2\$+1			;SAVE CC'S IN ODD BYTE 2\$
6381	026044	020405			CMP	R4,R5			;CHECK RESULTS
6382	026046	001004			BNE	3\$			
6383	026050	126767	177756	177755	CMP	2\$,2\$+1			;CHECK RESULT CC'S
6384	026056	001401			BEQ	+.4			
6385	026060	104000			3\$: HLT				;ERROR! EITHER RESULTS OR RESULT CC'S



6386  
6387 026062 005300  
6388 026064 005316  
6389 026066 020002  
6390 026070 001351  
6391  
6392  
6393  
6394  
6395  
6396  
6397 026072  
6398 026072 000004  
6399 026074 112737 000056 001202  
6400 026102 010703  
6401 026104 006702  
6402 026106 010304  
6403 026110 010316  
6404 026112 005216  
6405 026114 100002  
6406 026116 162716 000002  
6407 026122 071216  
6408 026124 103410  
6409 026126 102407  
6410 026130 001006  
6411 026132 100405  
6412 026134 005702  
6413 026136 001361  
6414 026140 010416  
6415 026142 020316  
6416 026144 001401  
6417 026146 104000  
6418  
6419  
6420  
6421  
6422  
6423  
6424  
6425  
6426  
6427  
6428  
6429  
6430  
6431  
6432  
6433  
6434  
6435  
6436  
6437  
6438  
6439  
6440  
6441

```

: DID NOT COMPARE
DEC RO : DECREMENT SHIFT COUNT
DEC (SP) : DECREMENT SHIFT COUNT FOR ASH (SP),R4
CMP RO,R2 : CHECK FOR MAX RIGHT SHIFT
BNE 1$

:*****
:TEST 56 DIVIDE AGAIN
: THE BELOW TEST CHECKS THE DIVIDE INSTRUCTION BY DIVIDING
: THE CURRENT PC BY ITSELF+1. THE QUOTIENT (IN R2) ALWAYS = 0,
: AND THE REMAINDER (IN R3) ALWAYS = THE CURRENT PC.
:*****
TST56:
SCOPE
MOV #56,0#STSTNM
DIV1: MOV PC,R3 : CURRENT PC IS LSH DIVIDEND
SXT R2 : EXTEND SIGN TO R2 (MSH DIVIDEND)
MOV R3,R4 : SAVE ORIGINAL DIVIDEND
MOV R3,(SP) : PUT ON STACK
INC (SP) : ADD 1 (WILL BE DIVISOR)
BPL 1$ : BRANCH IF POSITIVE
SUB #2,(SP) : MAKE DIVISOR 1 LESS THAN DIVIDEND
1$: DIV (SP),R2 : DIVIDE R2 BY (SP)
BCS 2$ : CHECK CONDITION CODES
BVS 2$
BNE 2$
BMI 2$
TST R2 : CHECK QUOTIENT (R2 = 0)
BNE DIV1
MOV R4,(SP) : GET ORIGINAL DIVISOR
CMP R3,(SP) : CHECK REMAINDER
BEQ .+4
2$: HLT : REPORT ERROR

```

```

: *
: * THIS SECTION OF THE MED TESTS EXERCISES CERTAIN SCRATCH
: * PAD REGISTERS USING MED READS AND WRITES. THEIR ORIGINAL
: * CONTENTS ARE RESTORED BUT:
: *
: * ***** IMPORTANT NOTE *****
: *
: * THE CONSOLE MUST NOT !!! BE USED DURING THESE MED
: * TESTS. NO INTERRUPTS OR TRAPS CAN BE ALLOWED EITHER
: * (THE T-BIT IS CLEARED FOR THESE TESTS)
: *
: * *****
: *
: * *****
: *TEST 57 CHECK MED IS ILLEGAL IN USER
: THE NEXT TWO TESTS BELOW CHECK TO SEE THAT THE "MED"
: (MAINTENANCE, EXAM, AND DEPOSIT) INSTRUCTION WILL EXECUTE
: WHEN IN KERNEL MODE AND THAT IT IS ILLEGAL IN
: USER MODE

```

```

6442
6443 026150
6444 026150 000004
6445 026152 112737 000057 001202
6446 026160 012737 000001 001324
6447 026166 013737 177776 002362
6448 026174 004737 054112
6449 026200 012700 000340
6450 026204 010037 000006
6451 026210 010037 000012
6452 026214 010037 000022
6453 026220 010037 000032
6454 026224 010037 000116
6455 026230 012737 140340 177776
6456 026236 012706 000700
6457 026242 010746
6458 026244 062716 000032
6459 026250 011637 000004
6460 026254 012637 000010
6461 026260 012701 177777
6462 026264 005000
6463 026266 076600
6464 026270 000041
6465 026272 104000
6466 026274 000407
6467 026276 005700
6468 026300 001401
6469 026302 104000
6470
6471 026304 010716
6472 026306 062716 000006
6473 026312 000002
6474 026314 012737 055064 000004
6475 026322 012737 054774 000010
6476
6477
6478
6479 026330
6480 026330 000004
6481 026332 112737 000060 001202
6482
6483 026340 005037 177776
6484 026344 000257
6485 026346 076600
6486 026350 000041
6487 026352 103403
6488 026354 102402
6489 026356 100401
6490 026360 001001
6491 026362 104000
6492
6493
6494
6495
6496
6497

*****
TST57:
SCOPE
MOV #57, @STSTNM
MOV #1, @STTIMES ; DO 1 ITERATION
MED1: MOV @PSW, @PSWHOL ; SAVE PSW
JSR PC, @CLRTBIT ; GO CLEAR "T-BIT" IF SET
MOV #340, R0 ; SET "VECTOR+2'S" UP FOR MED TESTS
MOV R0, @6
MOV R0, @12
MOV R0, @22
MOV R0, @32
MOV R0, @116
MOV #140340, @PSW ; GO TO USER MODE
MOV @USESTK, SP ; SETUP USER STACK PTR.
MOV PC, -(SP) ; GET CURRENT PC
ADD #25-, (SP)
MOV (SP), @ERRVEC ; SET ERROR TRAP VECTOR TO 25 BELOW
MOV (SP)+, @RESVEC ; LOAD RESERVED INST. TRAP VECTOR
MOV #-1, R1 ; LOAD R1 WITH A -1
CLR R0 ; CLEAR R0
MED ; TRY TO DO MAINT. EXAMINE
.WORD 041 ; MED READ CODE FOR R1
1$: HLT ; ERROR - MED INST. NOT ILLEGAL IN USER
BR 4$
2$: TST R0 ; IS R0 UNCHANGED?
BEQ 3$ ; BRANCH IF YES
HLT ; ERROR - MED INSTRUCTION WAS EXECUTED
BEFORE TRAPPING
3$: MOV PC, (SP) ; REPLACE RETURN PC WITH
ADD #45-, (SP) ; ADDRESS OF 4$ BELOW
RTI ; RETURN (TO 4$)
4$: MOV @ERPRT, @ERRVEC ; RESTORE ERROR TRAP VECTOR
MOV @RESERR, @RESVEC ; RESTORE RESERVED INST. TRAP VECTOR
*****
; TEST 60 CHECK MED DOESN'T AFFECT PSW
*****
TST60:
SCOPE
MOV #60, @STSTNM
MED0: CLR @PSW ; GO TO KERNEL MODE
CCC ; CLEAR CONDITION CODE BITS
MED ; DO MAINT. EXAMINE OF R1
.WORD 041 ; MED READ CODE FOR R1
BCS MEDHLT
BVS MEDHLT
BMI MEDHLT
BNE .+4
MEDHLT: HLT ; ERROR CC-BITS IN PSW AFFECTED BY MED
*****
; TEST 61 MED TEST - R/W DATA PATTERNS TO REGS
; THIS PARTICULAR MED TEST WRITES DATA PATTERNS
; TO THOSE INTERNAL REGS. WHICH CAN BE WRITTEN
; AND READ WITHOUT SPECIAL CONSIDERATIONS. REGISTERS

```



```

6498
6499
6500
6501
6502
6503 026364
6504 026364 000004
6505 026366 112737 000061 001202
6506 026374 012737 000340 177776
6507 026402 012701 002502
6508 026406 012737 125252 002476
6509 026414 111167 000036
6510 026420 112167 000054
6511 026424 111167 000006
6512 026430 112167 000030
6513 026434 076600
6514 026436 000000
6515 026440 010037 002472
6516 026444 010137 002474
6517 026450 013700 002476
6518 026454 076600
6519 026456 000000
6520 026460 005000
6521 026462 076600
6522 026464 000000
6523 026466 010037 002500
6524 026472 013700 002472
6525 026476 076600
6526 026500 000000
6527 026502 023737 002476 002500
6528 026510 001401
6529 026512 104000
6530
6531
6532
6533 026514 005137 002476
6534 026520 013701 002474
6535 026524 022737 125252 002476
6536 026532 001340
6537 026534 005711
6538 026536 001323
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553 026540

```

```

;* REQUIRING SPECIAL TESTS ARE TESTED IN LATER
;* MED TESTS.
;* TABLE II CONTAINS THE REGISTER ADDRESSES.
*****
TST61:
SCOPE
MOV8 #61, @STSTNM
MEDT1: MOV #340, @PSW ; KERNEL MODE-PRIORITY 7
MOV @TBL2, R1 ; INITIALIZE ADDRESS POINTER
1S: MOV #125252, @MEDTP2
MOV8 (R1), 11$ ; PUT WRITE CODE BY "WRITE-MED'S"
MOV8 (R1)+, 13$ ; AND POINT R1 TO READ CODE
MOV8 (R1), 10$ ; PUT READ CODE BY "READ-MED'S"
MOV8 (R1)+, 12$ ; R1 NOW POINTS TO NEXT REG.
2S: MED ; MED-READ THE INTERNAL REG.
10S: .WORD 0 ; MED-READ CODE
MOV RO, @MEDTPO ; SAVE ITS ORIGINAL CONTENTS
MOV R1, @MEDTP1 ; SAVE ADDR. PTR. VALUE
MOV @MEDTP2, RO ; LOAD RO WITH DATA TO BE WRITTEN
MED ; MED-WRITE THE TEST DATA
11S: .WORD 0 ; MED-WRITE CODE
CLR RO ; CLEAR RO
MED ; MED-READ THE DATA BACK
12S: .WORD 0 ; MED-READ CODE
MOV RO, @MEDTP3 ; SAVE DATA READ FOR COMPARISON
MOV @MEDTPO, RO ; LOAD ORIGINAL DATA IN RO
MED ; MED-WRITE ORG. DATA TO REG.
13S: .WORD 0 ; MED-WRITE CODE
CMP @MEDTP2, @MEDTP3 ; DID DATA READ=DATA WRITTEN?
BEQ 3S ; BRANCH IF YES
HLT ; INT. REG. READ BACK WRONG DATA
; MEDCODE WILL BE AT 12S
; DATA EXPECTED IS IN MEDTP2
; DATA RECEIVED IS IN MEDTP3
3S: COM @MEDTP2 ; CHANGE DATA PATTERN
MOV @MEDTP1, R1 ; RESTORE ADDR. POINTER
CMP #125252, @MEDTP2 ; BOTH DATA PATTERNS BEEN USED?
BNE 2S ; BRANCH IF NO
TST (R1) ; END OF ADDR. TABLE?
BNE 1S ; BRANCH IF NO

```

```

*****
TEST 62 MED TEST - CSP CONSTANTS CHECK
;*
;* THIS TEST CHECKS THE CONSTANT VALUES LOCATED
;* IN THE C SCRATCH PAD. THE CONSTANTS ARE READ
;* WITH A MED INSTRUCTION AND COMPARED TO THEIR
;* EXPECTED VALUE. THE ADDRESSES OF THESE CONSTANTS
;* AND THE VALUES EXPECTED ARE IN TABLE VII.
*****
TST62:

```

M11

```

6554 026540 000004          SCOPE
6555 026542 112737 000062 001202  MOVB      #62,2#STSTNM
6556
6557 026550 170000          MEDT10: CFCC          ;EXECUTE FLT. PT INST. SO FLT. PT.
6558                                     ;CONSTANTS ARE LOADED INTO CSP
6559 026552 012701 003034          MOV      #TBL7,R1      ;SETUP TABLE POINTER
6560 026556 012167 000006 10$:  MOV      (R1)+,1$      ;LOAD MED READ CODE AT 1$
6561 026562 001412          BEQ      11$          ;EXIT IF END OF TEST
6562 026564 005000          CLR      RO
6563 026566 076600          MED          ;READ INTERNAL CONTENTS INTO RO
6564 026570 000000 1$:      .WORD      0
6565 026572 020021          CMP      RO,(R1)+      ;DID I READ THE CONSTANT I EXPECTED
6566 026574 001770          BEQ      10$          ;BRANCH IF YES
6567 026576 016137 177776 002476  MOV      -2(R1),2#MEDTP2 ;SAVE CONSTANT VALUE EXPECTED
6568 026604 104000          HLT          ;CSP LOCATION HELD WRONG VALUE
6569                                     ;MEDCODE IS AT 1$
6570                                     ;DATA EXPECTED IS IN MEDTP2
6571                                     ;DATA RECEIVED IS IN RO OR THE LOC. ITSELF
6572 026606 000763          BR      10$          ;BRANCH BACK TO TEST NEXT CONSTANT
6573 026610 11$:

```

```

*****
*TEST 63      MED TEST - MICROBK CHECK OF MICRO-POINTS
*
*      THIS TEST USES THE MICROBREAK REGISTER AND THE
*      INFORMATION IN TABLE V TO CHECK THAT THE
*      CORRECT MED-FLOW IS ENTERED WHEN EACH
*      REGISTER IS ACCESSED BY A MED INSTRUCTION.
*      THE MICROBREAK REG. IS SETUP TO CAUSE A TRAP TO
*      LOC. 4 WHEN ITS CONTENTS EQUAL THE ADDRESS
*      OF THE MIRCOWORD BEING EXECUTED.
*
*      NOTE:  THE MICRO BREAK - TRAP-TO-4 CAPABILITY
*      IS TRIED AT THE BEGINNING OF THE TEST.
*      IF IT DOESN'T WORK, AN ERROR IS PRINTED
*      AND THE TEST IS SKIPPED
*****

```

```

6593 026610 000004          TST63:  SCOPE
6594 026610 112737 000063 001202  MOVB      #63,2#STSTNM
6595 026612 012737 000071 177770  MEDT11:  MOV      #SWB01,2#UBREAK ;LOAD MICROBK. REG. WITH AN MICRO ADDR.
6596 026620 012737 055172 000004  MOV      #BKROUT,2#4      ;LOAD ADDR. OF MICROBK. ROUTINE IN 4
6597 026626 005037 055200          CLR      2#BKFLAG        ;CLEAR MICROBK. TRAP FLAG
6598 026634 076600          MED
6599 026640 000022          RDWHAMI
6600 026642 042700 100001          BIC      #BIT15+BIT0,RO ;ENSURE LOG "CONTINUOUS" MODE
6601 026644 052700 001000          BIS      #BIT9,RO
6602 026650 000222          MED          ;MED-WRITE THE WHAMI REG TO
6603 026654 076600 100000          WRWHAMI ;ENABLE MICROBK-TRAP-TO-4
6604 026656 000144          MED
6605 026660 052700          RDFLAG
6606 026662 000144          BIS      #BIT15,RO
6607 026664 076600          MED          ;MED-WRITE THE FLAG REG TO
6608 026670 000344          WRFLAG ;ENABLE MICROBK TRAPPING
6609

```



```

6610 026674 000300 SWAB RO ; MICROBK TRAP SHOULD OCCUR ON SWAB
6611 026676 005737 055200 TST 2#BKFLAG ; DID TRAP TO 4 OCCUR?
6612 026702 001020 BNE 15 ; BRANCH IF YES
6613 026704 104000 HLT ; MICROBREAK TRAP DIDN'T WORK
6614 ; WHEN TRIED TO TRAP IN "SWAB" ROUTINE
6615 ; THEREFORE REST OF TEST IS SKIPPED
6616 026706 104401 026714 TYPE 65$ ; TYPE ASCIZ STRING
6617 026712 000413 BR 64$ ; GET OVER THE ASCIZ
6618 ; 65$: .ASCIZ <15><12>?SKIP TO NEXT TEST?<15><12>
6619 026742 64$: BR 50$ ; SKIP TO END OF TEST
6620 026742 000444
6621
6622
6623 026744 076600 15: MED ; READ THE LOG CUA REG.
6624 026746 000103 RDL CUA
6625 026750 042700 100007 BIC #100007,RO ; MASK OFF NON-ADDRESS BITS
6626 026754 020027 000710 CMP RO,#SWB1A ; WAS CORRECT MICRO ADDR. LOGGED?
6627 ; (MICRO ADDR. TRAPPED ON SHIFTED RIGHT)
6628 026760 001401 BEQ 16$ ; BRANCH IF YES
6629 026762 104000 HLT ; LOG CUA DID NOT CONTAIN MICRO
6630 026764 16$: ADDR. IT SHOULD HAVE CONTAINED AFTER
6631 ; MICROBREAKING ON "SWB01"
6632 026764 012701 002734 MOV #TBL5,R1 ; INITIALIZE TABLE PTR. (R1)
6633 026770 012702 002762 MOV #TBL6,R2
6634 026774 105711 25: TSTB (R1) ; IS OPERATION CODE = 0
6635 026776 001426 BEQ 50$ ; BRANCH IF YES, END OF TABLE
6636 027000 111167 000030 MOVB (R1),12$ ; IF NO, LOAD WRITE CODE AFTER MED
6637 027004 011237 177770 45: MOV (R2),2#UBREAK ; LOAD MICROBK REG. WITH MICROADDR.
6638 027010 005037 055200 CLR 2#BKFLAG ; CLEAR MICROBK TRAP-TO-4 FLAG
6639 027014 076600 MED
6640 027016 000144 RDL FLAG
6641 027020 052700 100000 BIS #BIT15,RO
6642 027024 076600 MED
6643 027026 000344 15$: WRFLAG ; MED WRITE TO FLAG REG TO
6644 027030 005000 CLR RO ; ENABLE MICROBK TRAPPING
6645 027032 076600 ; CLEAR RO IN CASE MICROBK DOESN'T OCCUR
6646 027034 000000 MED
6647 027036 005737 055200 12$: .WORD 0
6648 027042 001001 TST 2#BKFLAG ; DID WE TRAP-TO-4? (FLAG NOT = 0)
6649 027044 104000 HLT ; BRANCH IF YES TO NEXT ENTRY
6650 ; MICROBK. TRAP-TO-4 DID NOT OCCUR
6651 ; MEDCODE IS AT 12$
6652 ; MICROADDRESS IS AT ADDRESS CONTAINED IN R2)
6653 027046 105721 20$: TSTB (R1)+
6654 027050 005722 TST (R2)+
6655 027052 000750 BR 25 ; BRANCH BACK TO 25
6656
6657 027054 076600 50$: MED ; TEST IS OVER, DISABLE TRAPPING
6658 027056 000022 RDL WHAMI
6659 027060 042700 001000 BIC #BIT9,RO
6660 027064 076600 MED
6661 027066 000222 14$: WRWHAMI ; CLEAR THE WHAMI REG. TO
6662 027070 012737 055064 000004 MOV #ERPRT,2#4 ; DISABLE MICROBK. TRAP-TO-4
6663 ; RESTORE NORMAL ERROR ROUTINE
6664
6665 ;* THE FOLLOWING TESTS WILL BE EXECUTED WHEN RELOCATED

```

```

6666
6667
6668
6669
6670
6671
6672
6673
6674
6675
6676
6677 027076
6678 027076 000004
6679 027100 112737 000064 001202
6680 027106 010700
6681 027110 062700 000042
6682 027114 010701
6683 027116 062701 000005
6684 027122 010037 000004
6685 027126 076600
6686 027130 000022
6687 027132 052700 100001
6688 027136 076600
6689 027140 000222
6690 027142 005767 177755
6691
6692 027146 104000
6693 027150 000454
6694 027152 022626
6695 027154 012737 055064 000004
6696 027162 032737 000100 177766
6697
6698 027170 001001
6699 027172 104000
6700
6701
6702 027174 076600
6703 027176 000100
6704 027200 032700 100004
6705 027204 001001
6706 027206 104000
6707
6708 027210 010137 001536
6709 027214 004737 053724
6710 027220 005005
6711 027222 076600
6712 027224 000102
6713 027226 010037 002472
6714 027232 063737 001550 001540
6715 027240 020037 001540
6716
6717 027244 001401
6718 027246 005205
6719 027250 076600
6720 027252 000101
6721 027254 000300

```

```

;*
*****
*TEST 64 PHYSICAL ADDRESS & ODD ADDRESS ERROR LOGGING
* THIS TEST CHECKS THAT THE PROPER PHYSICAL ADDRESS BITS
* <17:00> ARE LOGGED UPON ERROR. THE ERROR IS CAUSED BY
* FORCING AN ODD ADDRESS TRAP. THE ERROR LOG MODE USED
* IS "LOG FIRST". ALSO, THE ODD ADDRESS ERROR BITS IN
* THE LOG JAM AND CPU ERROR REGISTER ARE CHECKED.
*****
TST64:
SCOPE
MOV8 #64,2#STSTNM
MOV PC,RO ;SETUP NEW PC & PSM FOR THE
ADD #2$--,RO ;ODD ADDRESS SERVICE ROUTINE
MOV PC,R1 ;GET CURRENT PC
ADD #1$+1--,R1 ;FORM ADDRESS OF 1$+1
1$: MOV RO,2#4
MED
RDLJAMI
BIS #BIT15+BIT0,RO ;SETUP "LOG FIRST" MODE
MED
WRAMI
JST 1$+1 ;DO ODD ADDRESS INSTRUCTION TO FORCE
;A JAMUPP & TRAP TO 4
;*** ODD ADDR. TRAP DID NOT OCCUR
;EXIT TEST
;RESTORE STACK
;RESTORE OLD PC
;WAS ODD ADDR. ERROR RECORDED BY
;THE CPU ERROR REGISTER?
;BRANCH IF YES
;*** CPU ERROR REG. DID NOT
;REPORT ODD ADDRESS ERROR
;READ THE LOG JAM REGISTER
;READ LOG JAM REG.
3$: MED
RDLJAM
BIT #BIT15+BIT2,RO ;WAS ODD ADDR. ERROR LOGGED BY LOG JAM
;BRANCH IF YES
;*** LOG JAM REG. DID NOT LOG
;ODD ADDRESS ERROR CORRECTLY
;STORE VIRTUAL ADDR. OF ODD ADDR. INSTRUCTION
;CONVERT IT TO AN 18-BIT PHYSICAL ADDR.
4$: MOV R1,2#VADR
JSR PC,2#CNVADR
CLR R5
MED
RDLPBA ;READ THE LOG PBA REGISTER
MOV RO,2#MEDTPO
ADD 2#FACTOR,2#PA1500 ;ADD RELOCATION FACTOR
CMP RO,2#PA1500 ;WERE BITS <15:00> OF THE PHYSICAL
;BUS ADDR. LOGGED CORRECTLY?
;BRANCH IF YES
5$: BEQ 5$
INC R5
MED ;READ THE LOG SERVICE REGISTER
RDLSERVICE
SWAB RO ;GET "PBA 17&16" DOWN TO BIT POSITION 0&1

```



```

6722 027256 042700 177774      BIC      #177774,R0
6723 027262 010037 002476      MOV      R0,#MEDTP2
6724 027266 020037 001542      CMP      R0,#PA1716      ;PBA <17:16> LOGGED CORRECTLY?
6725 027272 001002          BNE      11$              ;BRANCH IF YES
6726 027274 005705          TST      R5
6727 027276 001401          BEQ      10$
6728 027300          11$:
6729 027300 104000          HLT
6730
6731
6732
6733 027302 012737 055064 000004 10$:  MOV      #ERPRT,#4      ;*** PHYSICAL BUS ADDR. <17:00>
6734 027310 076600          MED
6735 027312 000022          RDWHAMI      ;NOT LOGGED CORRECTLY WHEN
6736 027314 042700 100001          BIC      #BIT15+BIT0,R0  ;ODD ADDRESS TRAP OCCURRED
6737 027320 076600          MED
6738 027322 000222          WRWHAMI      ;EXPECTED PBA IS IN "PA1716" & "PA1500"
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759 027324          ;*
6760 027324 000004          ;*
6761 027326 112737 000065 001202          ;*THE FOLLOWING TESTS WILL NOT BE EXECUTED WHEN RELOCATED
6762 027334 012737 000001 001324          ;*
6763
6764
6765
6766
6767
6768
6769
6770
6771
6772 027366 012701 002360          ;*****
6773 027372 005711          ;*TEST 65      CHECK DISABLE PARITY ERROR TRAP
6774 027374 012737 000100 177746          ;*THIS TEST CHECKS THAT PARITY ERROR TRAPS TO LOCATION 114
6775 027402 012711 125252          ;*ARE DISABLED WHEN BIT0 OF THE CACHE CONTROL REGISTER IS
6776 027406 012737 000001 177746          ;*SET (=1). A TRAP TO 114 SHOULD NOT OCCUR AND ERROR
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819
6820
6821
6822
6823
6824
6825
6826
6827
6828
6829
6830
6831
6832
6833
6834
6835
6836
6837
6838
6839
6840
6841
6842
6843
6844
6845
6846
6847
6848
6849
6850
6851
6852
6853
6854
6855
6856
6857
6858
6859
6860
6861
6862
6863
6864
6865
6866
6867
6868
6869
6870
6871
6872
6873
6874
6875
6876
6877
6878
6879
6880
6881
6882
6883
6884
6885
6886
6887
6888
6889
6890
6891
6892
6893
6894
6895
6896
6897
6898
6899
6900
6901
6902
6903
6904
6905
6906
6907
6908
6909
6910
6911
6912
6913
6914
6915
6916
6917
6918
6919
6920
6921
6922
6923
6924
6925
6926
6927
6928
6929
6930
6931
6932
6933
6934
6935
6936
6937
6938
6939
6940
6941
6942
6943
6944
6945
6946
6947
6948
6949
6950
6951
6952
6953
6954
6955
6956
6957
6958
6959
6960
6961
6962
6963
6964
6965
6966
6967
6968
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995
6996
6997
6998
6999
7000

```





```

6834 027564 000004 SCOPE
6835 027566 112737 000066 001202 MOVB #66,2#STSTNM
6836 027574 012701 002360 MOV #TLOC1,R1 ;GET POINTER TO TEST LOCATION
6837 027600 005711 TST (R1) ;MAKE IT A HIT
6838 027602 012737 000100 177746 MOV #WMP,2#CCR ;SET WRITE WRONG PARITY BIT
6839 027610 012711 125252 MOV #125252 (R1) ;WRITE TO TEST LOC. WITH WRONG PARITY
6840 027614 042737 000100 177746 BIC #WMP,2#CCR ;CLEAR WMP
6841 027622 012737 027650 000114 MOV #15,2#114 ;SETUP NEW TEST HANDLER AT PARITY VECTOR
6842 027630 005767 152524 TST TLOC1 ;READ TEST LOC. TO FORCE PARITY ERROR
6843 027634 012700 000200 MOV #200,R0
6844 027640 076600 MED ;CLEAN UP THE CACHE
6845 027642 000352 352 ;INITIALIZATION CODE
6846 027644 104000 HLT ;*** PARITY ERROR DID NOT CAUSE TRAP
6847 027646 000405 BR 25 ;BRANCH TO 25
6848 027650 012700 000200 15: MOV #200,R0
6849 027654 076600 MED ;CLEAN UP CACHE WITH INITIALIZATION CODE
6850 027656 000352 352
6851 027660 022626 CMP (SP)+,(SP)+ ;CLEAN UP STACK
6852 027662 022737 000340 177744 25: CMP #000340,2#MEMERR ;WERE PARITY ERROR BITS (5,6,7) SET
6853 ;AND CPU ABORT BIT (15) LEFT CLEAR
6854 ;IN MEMORY ERROR REGISTER?
6855 027670 001401 BEQ 35 ;BRANCH IF YES
6856 027672 104000 HLT ;*** MEMORY ERROR REGISTER BITS
6857 ;WERE SET INCORRECTLY
6858 ;PARITY ERROR BITS SHOULD BE SET
6859 027674 012737 054414 000114 35: MOV #.PARSRV,2#114 ;RESTORE OLD PARITY HANDLER PC
6860
6861 ;*****
6862 ;*TEST 67 CHECK UNIBUS TIMEOUT, ODD ADDRESS AND LOG CONTINUOUS MODE
6863
6864 ;*THIS TEST CHECKS THAT THE "UNIBUS TIMEOUT" BIT (BIT4)
6865 ;*GETS SET IN THE CPU ERROR REGISTER WHEN A TIMEOUT OCCURS.
6866 ;*A TIMEOUT TRAP IS FORCED BY REFERENCING BUS ADDRESS 760000.
6867 ;*THEN AN ODD ADDRESS ERROR IS FORCED AND IT
6868 ;*IS CHECKED IF ONLY BIT (6)-ODD ADDRESS ERROR IS SET
6869 ;*(IN CPUERR). THIS CHECKS THAT THE ERROR LOG IS
6870 ;*CONTINUOUSLY UPDATED IN THE "LOG CONTINUOUS" MODE.
6871 ;*****
6872 027702 †ST67:
6873 027702 000004 SCOPE
6874 027704 112737 000067 001202 MOVB #67,2#STSTNM
6875 027712 012737 027726 000004 MOV #15,2#4 ;SETUP NEW PC
6876 027720 005737 160000 TST 2#160000 ;FORCE A TIMEOUT TRAP TO 4 BY
6877 ;REFERENCING NON-EXISTENT ADDRESS
6878 027724 000452 BR 65
6879 027726 022626 15: CMP (SP)+,(SP)+ ;RESTORE STACK
6880 027730 012737 055064 000004 MOV #ERPR1,2#4 ;RESTORE OLD PC
6881 027736 022737 000020 177766 CMP #BIT4,2#CPUERR ;DID "UNIBUS TIMEOUT" BIT IN CPU ERROR
6882 ;REGISTER GET SET?
6883 027744 001401 BEQ 25 ;BRANCH IF YES
6884 027746 104000 HLT ;*** "UNIBUS TIMEOUT" BIT (BIT4) IN CPU
6885 ;ERROR REG. DID NOT SET WHEN A
6886 ;TIMEOUT WAS FORCED
6887 027750 076600 25: MED ;READ THE LOG JAM REG.
6888 027752 000100 RDLJAM
6889 027754 022700 021200 CMP #BIT13+BIT9+BIT7,R0 ;DID "UNIBUS TIMEOUT" BIT (BIT7) SET?

```

# F12

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 148  
T67 CHECK UNIBUS TIMEOUT, ODD ADDRESS AND LOG CONTINUOUS MODE

6890  
6891 027760 001401  
6892 027762 104000  
6893

BEQ 35  
HLT

:BIT 9= POWER STATUS, ALWAYS SET  
:BRANCH IF YES  
:\*\*\* "UNIBUS TIMEOUT" BIT (BIT7)  
:DID NOT SET IN LOG JAM REGISTER



```

6894                                     ;WHEN UNIBUS TIMEOUT WAS FORCED
6895 027764 076600 3S: MED ;READ LOG PBA
6896 027766 000102 RDLPBA
6897 027770 020027 160000 CMP RD,#160000 ;WAS PHYS BA LOGGED CORRECTLY?
6898 027774 001401 BEQ 5S
6899 027776 104000 HLT ;PHYSICAL BUS ADDRESS WAS
6900 ;LOGGED WRONG ON A UNIBUS
6901 ;TIMEOUT
6902 ;LOG PBA SHOULD HOLD ADDR. 160000
6903 030000 012737 030014 000004 5S: MOV #4S,2#4 ;SET UP PC FOR ODD ADDRESS
6904 030006 005767 177753 TST 3S+1 ;FORCE ODD ADDRESS ERROR
6905 030012 000417 BR 6S
6906 030014 022626 4S: CMP (SP)+,(SP)+ ;RESTORE STACK
6907 030016 012737 055064 000004 MOV #ERPRT,2#4
6908 030024 022737 000100 177766 CMP #BIT6,2#CPUERR ;ODD ADDR. BUT SET 3
6909 030032 001401 BEQ 7S
6910 030034 104000 HLT ;ODD ADDRESS BIT WAS
6911 ;NOT SET IN THE CPU
6912 ;ERROR REGISTER. IN LOG
6913 ;CONTINUOUS MADE THE
6914 ;LATEST ERROR SHOULD
6915 ;BE LOGGED
6916 030036 076600 7S: MED ;READ LOG JAM REG.
6917 030040 000100 RDLJAM
6918 030042 032700 000004 BIT #BIT2,RO
6919 030046 001001 BNE 6S ;ODD ADR. BIT SET IN
6920 030050 104000 HLT ;LOG JAM?
6921 ;ODD ADDRESS BIT WAS
6922 ;NOT SET IN THE LOG
6923 ;JAM REGISTER ON A
6924 030052 076600 6S: MED ;ODD ADDRESS ERROR
6925 030054 000104 RDLFGINT ;CHECK IF LAST INTERRUPT VECTOR
6926 030056 120027 000004 CMPB RO,#4 ;WAS LOGGED?
6927 030062 001401 BEQ 8S
6928 030064 104000 HLT ;LAST ERROR VECTOR WAS NOT LOGGED
6929 ;LOW BYTE OF LOG FLAG/INT REG.
6930 ;SHOULD CONTAIN LAST VEC. =004
6931 030066 012737 055064 000004 8S: MOV #ERPRT,2#4
6932
6933
6934
6935 ;*****
6936 ;*TEST 70 CHECK ILLEGAL INTERNAL ADDRESS TRAP
6937
6938 ;*THIS TEST CHECKS THAT A TRAP OCCURS UPON REFERENCING AN
6939 ;*ILLEGAL INTERNAL ADDRESS AND THAT "ILLEGAL INTERNAL ADDRESS"
6940 ;*BIT (BIT0) OF THE CPU ERROR REGISTER AND BITS OF LOG JAM
6941 ;*REGISTER GET SET. IT ALSO CHECKS IF THE INTERRUPT VECTOR
6942 ;*(4) IS SAVED AS THE "LAST INTERRUPT VECTOR" IN THE LOG
6943 ;*FLAG/INTERRUPT REG.
6944 ;*****
6945 †ST70:
6946 030074 000004 SCOPE
6947 030076 112737 000070 001202 MOVB #70,2#STSTNM
6948 030104 012737 030126 000004 MOV #1S,2#4 ;SETUP NEW HANDLER PC
6949 030112 005037 177746 CLR 2#CCR

```

# H12

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 150  
DQKDC.A.P11 07-FEB-77 09:58 T70 CHECK ILLEGAL INTERNAL ADDRESS TRAP

```
6950 030116 012707 177746      MOV      #CCR,PC      ; ILLEGAL INTERNAL ADDRESS TRAP SHOULD OCCUR
6951 030122 104000      HLT
6952
6953 030124 000417      BR       3$          ; *** ILLEGAL INTERNAL ADDRESS
6954 030126 022626      1$:  CMP      (SP)+,(SP)+ ; DID NOT RESULT IN A TRAP
6955 030130 012737 055064 000004      MOV      #ERPR1,2#4  ; BRANCH TO EXIT IF NO TRAP
6956 030136 032737 000001 177766      BIT      #BIT0,2#CPUERR ; RESTORE STACK
6957
6958 030144 001001      BNE     2$          ; RESTORE OLD HANDLER PC
6959 030146 104000      HLT
6960
6961 030150 076600      2$:  MED
6962 030152 000100      RDLJAM
6963 030154 032700 000040      BIT      #BIT5,RO    ; DID "ILLEGAL INTERNAL ADDRESS" BIT (0)
6964
6965 030160 001001      BNE     3$          ; IN CPU ERROR REGISTER GET SET?
6966 030162 104000      HLT
6967
6968 030164      3$:
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
```

```
::*****
; *TEST 71 CHECK LOG SERVICE & MEMERR LOGS LO-HI BYTE & TAG BITS, IN ABORT MODE
; THIS TEST CHECKS THAT "LO BYTE PARITY" AND "TAG PARITY"
; BITS CAN SET IN "LOG SERVICE" REGISTERS. IT IS ALSO
; CHECKED THAT THE PROPER TAG AND DATA BITS GET STORED
; IN THE "LOG CACHE DATA, "LOG CACHE TAG/CPU" AND THE
; "MEMORY ADDRESS REGISTER" WHEN A PARITY ERROR IS
; FORCED.
; IT IS CHECKED IF THE INSTRUCTION WAS ABORTED AND THE
; LOG FLAG/INTERRUPT REGISTER LOGGED THE LAST INTERRUPT
; VECTOR.
```

```
::*****
; *TEST 71:
6983 030164      SCOPE
6984 030164 000004      MOV      #71,2#STSTNM
6985 030166 112737 000071 001202
6986
6987 030174 012737 000201 177746      MOV      #DPTRP+PABORT,2#CCR ; DISABLE PARITY TRAPS (CACHE)
6988 030202 005067 152264      CLR      MEDTPO
6989 030206 012701 002360      MOV      #TLOC1,R1      ; GET POINTER TO TEST LOC.
6990 030212 012711 111000      MOV      #111000,(R1)
6991 030216 005711      TST     (R1)          ; MAKE IT A HIT
6992 030220 052737 000100 177746      BIS      #WMP,2#CCR     ; WRITE WRONG PARITY SET
6993 030226 012711 000252      MOV      #252,(R1)     ; WRITE TEST LOCATION
6994
6995 030232 042737 000100 177746      BIC      #WMP,2#CCR    ; WITH WRONG PARITY
6996 030240 076600      MED
6997 030242 000022      RDWHAMI ; CLEAR WMP
6998 030244 052700 100001      BIS      #BIT15+BIT0,RO ; ENABLE "LOG FIRST" MODE AND
6999 030250 076600      MED ; ERROR LOGGING
7000 030252 000222      WRWHAMI
7001 030254 042737 000001 177746      BIC      #DPTRP,2#CCR  ; ENABLE CACHE PARITY TRAPS
7002 030262 012737 030310 000114      MOV      #PTRP1,2#114  ; NEW PARITY TRAP SERVICE
7003 030270 016767 152064 152174      MOV      TLOC1,MEDTPO ; READ TEST LOC, FORCE PARITY ERROR
7004 030276 012700 000200      MOV      #200,RO
7005 030302 076600      MED ; CLEAN UP THE CACHE
```



7006	030304	000352				352				; INITIALIZATION CODE
7007	030306	104000				HLT				; *** CACHE PARITY ERROR TRAP
7008										; DID NOT OCCUR WHEN
7009										; TEST LOC WITH BAD PARITY
7010										; WAS READ
7011										
7012										; ENTER HERE IF PARITY TRAP OCCURRED
7013	030310	012700	000200		PTRP1:	MOV	#200,R0			
7014	030314	076600				MED				; CLEAN UP THE CACHE
7015	030316	000352				352				; INITIALIZATION CODE
7016	030320	012737	000001	177746		MOV	#DPTRP,#CCR			; DISABLE CACHE PARITY ERROR TRAPS
7017	030326	012737	054414	000114		MOV	#.PARSAV,#114			; REESTABLISH OLD SERVICE VECTORS
7018	030334	022626				CMP	(SP)+,(SP)+			
7019	030336	005767	152130			TST	MEDTP0			; WAS THE INSTRUCTION ABORTED ON
7020										; CACHE PARITY ERROR (ABORT MODE)?
7021	030342	001401				BEQ	15			; YES
7022	030344	104000				HLT				; INSTRUCTION HAVING CACHE PARITY
7023										; ERROR WAS NOT ABORTED, IN THE
7024										; CACHE ABORT MODE.
7025	030346	076600			15:	MED				; READ THE "LOG SERVICE" REGISTER
7026	030350	000101				RDL	SERVICE			
7027	030352	042700	177435			BIC	#177435,R0			; MASK ALL BITS BUT LO HI TAG CACHE BITS
7028	030356	022700	000342			CMP	#342,R0			; LO HI TAG, CACHE PARITY BITS SET? IN "SERVICE"
7029	030362	001401				BEQ	25			; YES
7030	030364	104000				HLT				; *** "LO BYTE" PARITY ERROR
7031										; AND "TAG" PARITY ERROR BITS
7032										; WERE NOT LOGGED CORRECTLY IN "LOG
7033										; SERVICE" REGISTER, WHEN PARITY
7034										; ERROR TRAP WAS FORCED.
7035										; CLEAR BITS ARE ACTIVE.
7036	030366	022737	100340	177744	25:	CMP	#BIT7+BIT6+BITS+BIT15,#MEMERR			; DID "LO BYTE" AND "TAG"
7037										; PARITY ERROR BITS SET IN
7038										; THE MEMORY ERROR REGISTER?
7039	030374	001401				BEQ	35			; YES
7040	030376	104000				HLT				; *** "LO BYTE" AND "TAG" PARITY
7041										; ERROR BITS DID NOT SET
7042										; CORRECTLY IN THE MEMORY
7043										; ERROR REGISTER
7044	030400	076600			35:	MED				; READ "LOG PBA" REGISTER
7045	030402	000102				RDL	PBA			
7046	030404	020027	002360			CMP	R0,#TLOC1			; DID "LOG PBA" CONTAIN CORRECT
7047										; PHYSICAL BUS ADDRESS-WHERE
7048										; THE PARITY ERROR OCCURRED?
7049	030410	001401				BEQ	45			; YES
7050	030412	104000				HLT				; *** PHYSICAL BUS ADDRESS
7051										; (WHERE PARITY ERROR OCCURRED)
7052										; WAS NOT LOGGED CORRECTLY
7053										; WHEN CACHE PARITY ERROR WAS FORCED
7054										; LOG PBA SHOULD CONTAIN ADDR. OF "TLOC1"
7055	030414	076600			45:	MED				
7056	030416	000107				RDL	TAG			; READ "LOG CACHE TAG" REGISTER
7057	030420	010002				MOV	R0,R2			
7058	030422	000302				SWAB	R2			
7059	030424	012701	002360			MOV	#TLOC1,R1			; SHIFT RIGHT (3 TIMES) THE 16 BIT
7060	030430	000301				SWAB	R1			
7061	030432	106201				ASRB	R1			; PHYSICAL BUS ADDRESS OF THE

```

7062 030434 106201 ASRB R1 ;TEST LOCATION
7063 030436 106201 ASRB R1
7064 030440 052701 000200 BIS #BIT7,R1 ;FUDGE TAG BIT
7065 030444 120102 CMPB R1,R2 ;WAS THE CORRECT TAG LOGGED?
7066 030446 001405 BEQ SS ;YES
7067 030450 010167 152016 MOV R1,MEDTPO ;EXPECTED TAG
7068 030454 010267 152014 MOV R2,MEDTPI ;TAG RECEIVED
7069 030460 104000 HLT ;TAG BITS WERE NOT LOGGED
7070 ;CORRECTLY, WHEN CACHE
7071 ;PARITY ERROR WAS FORCED
7072 ;EXPECTED TAG IS IN MEDTPO
7073 ;TAG RECEIVED IS IN MEDTPI
7074 030462 076600 5S: MED ;READ CACHE DATA
7075 030464 000106 RDLDATA
7076 030466 020027 000252 CMP RO,#252 ;CACHE DATA LOGGED CORRECTLY?
7077 030472 001401 BEQ 6S
7078 030474 104000 HLT ;CACHE DATA WAS NOT LOGGED
7079 ;CORRECTLY, LOG DATA REG.
7080 ;SHOULD CONTAIN DATA = 252
7081 ;RETURN TO "LOG CONTINUOUS" MODE
7082 030476 076600 6S: MED
7083 030500 000022 RDLWHAMI
7084 030502 042700 100001 BIC #BIT15+BIT0,RO
7085 030506 076600 MED
7086 030510 000222 WRWHAMI
7087 030512 012737 030524 000004 MOV #7S,#4
7088 030520 005737 160000 TST #160000 ;INTERRUPT THRU 4 SHOULD LOAD "FLAG"
7089 030524 022626 7S: CMP (SP)+,(SP)+
7090 030526 012737 055064 000004 MOV #ERPRT,#4 ;RESTORE LOC. 4
7091 030534 076600 MED ;READ LOG FLAG/INTERRUPT REGISTER
7092 030536 000104 RDLFGINT
7093 030540 120027 000114 CMPB RO,#114 ;DID LO BYTE CONTAIN VECTOR 114?
7094 030544 001401 BEQ 8S
7095 030546 104000 HLT ;LAST INTERRUPT VECTOR WAS NOT
7096 ;LOGGED CORRECTLY IN FLAG REGISTER
7097 ;WHEN A CACHE PARITY ERROR WAS
7098 ;FORCED.
7099 ;LOW BYTE OF LOG FLAG/INT REG.
7100 ;SHOULD BE LAST VEC. = 114
7101 ;ANY ERROR CALL DURING TEST MIGHT CAUSE
7102 ;LAST VECTOR TO LOOK WRONG
7103 030550 8S:
7104
7105
7106 ;*****
7107 ;*TEST 72 CHECK "LOG FIRST" MODE OF ERROR LOGGING
7108 ;*THIS TEST CHECKS THE "LOG FIRST" MODE OF ERROR LOGGING.
7109 ;*THE "LOG FIRST" MODE IS ENABLED. THEN A TIME-OUT TRAP
7110 ;*IS FORCED. BIT 4 OF CPU ERROR REGISTER SHOULD BE SET.
7111 ;*THEN AN ODD ADDRESS TRAP IS FORCED. HOWEVER, THIS
7112 ;*TIME THE ERROR SHOULD NOT BE LOGGED; BIT 6 (ODD
7113 ;*ADDRESS) SHOULD NOT BE SET BECAUSE THE ERROR LOG
7114 ;*IS LOCKED UP AFTER THE FIRST ERROR.
7115
7116 ;*THEN, THE ERROR LOG IS ENABLED (BY SETTING BIT 0 OF
7117 ;*WHAMI). AN ODD ADDRESS ERROR IS FORCED AGAIN AND IT IS

```



```

7118                                     ;*CHECKED THAT THIS TIME THE ERROR IS LOGGED, (BIT 6-ODD
7119                                     ;*ADDRESS SHOULD BE SET IN CPU ERROR REGISTER).
7120                                     ;*****
7121 030550                                †ST72:
7122 030550 000004                        SCOPE
7123 030552 112737 000072 001202        MOVB    #72,2#STSTNM
7124
7125 030560 076600                        MED
7126 030562 000022                        RDWHAMI
7127 030564 052700 100001                BIS     #BIT15+BIT0,RO ;SET UP "LOG FIRST MODE
7128 030570 076600                        MED
7129 030572 000222                        WRWHAMI
7130 030574 012737 030610 000004        MOV     #15,2#4 ;SET UP NEW PC FOR TIMEOUT
7131 030602 005737 160000                TST    2#160000 ;FORCE A TIMEOUT
7132 030606 000454                        BR     5$ ;SKIP TEST IF NO TIMEOUT
7133
7134 030610 022626                        1$:    CMP    (SP)+,(SP)+ ;RESTORE STACK
7135                                     ;BIT 4 OF CPU ERROR REGISTER
7136                                     ;SHOULD HAVE SET
7137 030612 012737 030626 000004        MOV     #25,2#4 ;SET UP NEW PC FOR ODD ADDRESS
7138 030620 005767 177765                TST    1$+1 ;FORCE ODD ADDRESS TRAP
7139 030624 000445                        BR     5$ ;SKIP TEST IF NO ODD ADDRESS TRAP
7140
7141 030626 022626                        2$:    CMP    (SP)+,(SP)+ ;RESTORE STACK
7142 030630 012737 055064 000004        MOV     #ERPRT,2#4
7143 030636 022737 000020 177766        CMP    #BIT4,2#CPUERR ;"TIMEOUT" BIT SHOULD BE STILL
7144                                     ;SET, CHECK?
7145 030644 001402                        BEQ    3$
7146 030646 104000                        HLT
7147                                     ;*** SECOND ERROR (ODD ADDRESS)
7148                                     ;UPDATED THE ERROR LOG IN
7149                                     ;THE LOG FIRST MODE. BIT 4
7150                                     ;(UNIBUS TIMEOUT) SHOULD BE
7151                                     ;STILL SET FROM THE FIRST
7152 030650 000433                        BR     5$ ;ERROR
7153 030652 076600                        3$:    MED ;SKIP THE REST
7154 030654 000100                        RDJAM ;READ LOG JAM REG.
7155 030656 032700 100004                BIT    #BIT2+BIT15,RO ;CHECK THAT ODD ADRES ERROR BITS NOT
7156 030662 001401                        BEQ    6$ ;SET IN LOG JAM. NOTE LOG FIRST
7157                                     ;MODE SHOULD INHIBIT FURTHER
7158                                     ;ERROR LOGGING
7159 030664 104000                        HLT ;ODD ADDRESS ERROR BITS GOT SET IN LOG JAM
7160                                     ;THEY SHOULD NOT BE SINCE LOG FIRST MODE
7161                                     ;INHIBITS ERROR LOGGING AFTER THE FIRST ERROR
7162 030666 076600                        6$:    MED
7163 030670 000022                        RDWHAMI
7164 030672 052700 100001                BIS     #BIT15+BIT0,RO ;ENABLE ERROR LOG AGAIN IN
7165                                     ;LOG FIRST MODE
7166 030676 076600                        MED
7167 030700 000222                        WRWHAMI
7168 030702 012737 030716 000004        MOV     #4$,2#4 ;SET UP NEW PC
7169 030710 005767 177737                TST    3$+1 ;FORCE ODD ADDRESS TRAP
7170 030714 000411                        BR     5$ ;SKIP IF NO TRAP
7171 030716 022626                        4$:    CMP    (SP)+,(SP)+ ;RESTORE STACK
7172                                     ;RESTORE OLD PC(4)
7173 030720 012737 055064 000004        MOV     #ERPRT,2#4

```

```

7174 030726 022737 000100 177766      CMP      #BIT6,#CPUERR ;THE ERROR LOG FROM PREVIOUS
7175                                     ;ERROR SHOULD BE OVER WRITTEN.
7176                                     ;ODD ADDRESS BIT SHOULD
7177                                     ;BE SET, BECAUSE THE ERROR
7178 030734 001401                      BEQ      SS          ;LOG WAS ENABLED.
7179                                     ;OK, IF YES
7180 030736 104000                      HLT                                     ;THE ERROR LOG WAS NOT UPDATED
7181                                     ;(UPON AN ODD ADDRESS ERROR)
7182                                     ;AFTER THE LOG WAS ENABLED.
7183                                     ;AT THIS FORMAT BIT 6 OF
7184                                     ;CPU ERROR REGISTER SHOULD
7185                                     ;BE SET. IT WAS NOT.
7186 030740 012737 055064 000004 5S:    MOV      #ERPRT,#4    ;RESTORE OLD PC(4)
7187 030746 076600                      MED
7188 030750 000022                      RDLWHAMI
7189 030752 042700 100001              BIC      #BIT15+BIT0,RO
7190 030756 076600                      MED
7191 030760 000222                      RDLWHAMI ;PUT THE LOGGING BACK INTO
7192                                     ;"CONTINUOUS" MODE
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215
7216
7217
7218
7219
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229

```

```

*****
;TEST 73 CHECK LAST INTERRUPT VECTOR IS LOGGED IN FLAG REG.
*****

```

```

TST73:
SCOPE
MOV      #73,#STSTM
MOV      #1,#STIMES ;DO 1 ITERATION
MOV      #15,#30 ;LOAD LOC. 30 WITH 15 (EMT VEC.)
EMT ;FIRST INTERRUPT-EMT
15: CMP      (SP)+,(SP)+ ;CLEAN UP STACK
MOV      #ERROR,#30 ;RESTORE LOC. 30
MOV      #25,#4 ;LOAD LOC. 4 WITH 25
TST      #160000 ;SECOND INTERRUPT-SHOULD LOAD LOG FLAG REG.
25: CMP      (SP)+,(SP)+ ;CLEAN UP STACK
MOV      #ERPRT,#4 ;RESTORE LOC. 4
MED ;READ LOG FLAG/INT REG.
RDLFGINT
21: CMPB    RO,#30 ;WAS 30 LOGGED AS LAST INTRPT. VEC?
BEQ      35 ;BRANCH IF YES
HLT ;LOG FLAG/INT REG DID NOT LOG VECTOR
;LOW BYTE OF LOG FLAG/INT REG.
;SHOULD BE LAST VEC. = 30
35: MOV      #65,#20 ;LOAD LOC. 20 WITH 65
IOT ;FIRST INTERRUPT - IOT
65: CMP      (SP)+,(SP)+ ;CLEAN UP STACK
MOV      #SCOPE,#20 ;RESTORE CONTENTS OF LOC 20
MOV      #45,#4 ;LOAD LOC. 4 WITH 45
TST      #160000 ;INTERRUPT SHOULD LOAD LOG FLAG REG.
45: CMP      (SP)+,(SP)+ ;CLEAN UP STACK
MOV      #ERPRT,#4 ;RESTORE LOC. 4
MED ;READ LOG FLAG/INT REG.
RDLFGINT
22: CMPB    #20,RO ;WAS 20 LOGGED AS LAST INTRPT VEC?
BEQ      55 ;BRANCH IF YES
HLT ;LOG FLAG/INT REG. DID NOT LOG VECTOR

```



:LOW BYTE OF LOG FLAG/INT REG.  
:SHOULD BE LAST VEC. = 20  
:ANY ERROR CALL DURING THE TEST MIGHT  
:CAUSE LAST VECTOR TO BE WRONG

7230  
7231  
7232 031134  
7233  
7234  
7235  
7236  
7237  
7238  
7239 031134 013746 002362  
7240 031140 011600  
7241 031142 042700 000020  
7242 031146 052700 000340  
7243 031152 010037 000006  
7244 031156 010037 000012  
7245 031162 010037 000032  
7246 031166 010037 000116  
7247 031172 005037 000022  
7248 031176 010746  
7249 031200 062716 000006  
7250 031204 000002  
7251 031206 012706 000700  
7252  
7253  
7254  
7255 031212  
7256 031212 000004  
7257 031214 112737 000074 001202  
7258 031222 032737 140000 177776  
7259 031230 001535  
7260 031232 010746  
7261 031234 062716 000120  
7262 031240 012637 000250  
7263 031244 005046  
7264 031246 010603  
7265 031250 010346  
7266 031252 105737 001545  
7267 031256 001411  
7268 031260 013737 177640 177654  
7269 031266 012737 006006 177614  
7270 031274 062706 140000  
7271 031300 000240  
7272 031302 010746  
7273 031304 062716 000024  
7274 031310 012637 000020  
7275 031314 000004  
7276 031316 005266 000002  
7277 031322 001417  
7278 031324 104000  
7279 031326 000415  
7280 031330 000240  
7281 031332 006506  
7282 031334 006536  
7283 031336 006576 000000  
7284 031342 000240  
7285 031344 001367

MEDEX: MOV @PSWHOL, -(SP) ; PUSH OLD PSW ON STACK  
MOV (SP), R0 ; RESET "VECTOR + 2'S"  
BIC #BIT4, R0  
BIS #PR7, R0  
MOV R0, @#6  
MOV R0, @#12  
MOV R0, @#32  
MOV R0, @#116  
CLR @#22  
MOV PC, -(SP) ; PUSH CURRENT PC ON STACK  
ADD #6, (SP) ; ADD OFFSET TO PC  
RTI ; RESTORE ORIGINAL PSW AND CONTINUE  
MOV #USESTK, SP ; SET STACK POINTER

\*\*\*\*\*  
:TEST 74 CHECK MFPI/MTPI INSTRUCTIONS  
\*\*\*\*\*  
TST74:

SCOPE  
MPI: MOVB #74, @#STSTNM  
BIT #UM, @#PSW ; KERNEL MODE?  
BEQ ENDCP ; IF YES, EXIT TEST  
MOV PC, -(SP)  
ADD #5, -(SP)  
MOV (SP)+, @#MMVEC ; SET MEM MGMT ABORT VECTOR  
CLR -(SP) ; CLEAR CHECK WORD  
MOV SP, R3  
MOV R3, -(SP) ; PUT ADDRESS OF CHECK WORD ON THE STACK  
TSTB @#MMON ; CHECK IF MEM MGMT IS ENABLED  
BEQ IS ; BRANCH IF OFF  
MOV @#UIPAR0, @#UIPAR6 ; SET UP USER PAGE ADDR. REG.  
MOV #6006, @#UIPDR6 ; SET USER PAGE DESC REG R/W UP 6 PAGES  
IOS: ADD #140000, SP ; SET CURRENT MODE'S STACK POINTER  
IS: MOV PC, -(SP)  
ADD #3, -(SP)  
MOV (SP)+, @#IOTVEC ; SET IOT TRAP VECTOR  
IOT ; TRAP TO 3\$ BELOW  
INC 2(SP) ; INCREMENT CHECK WORD  
BEQ 6\$  
4\$: HLT ; ERROR! MFPI, MTPI FAILURE-FOR BETTER  
BR 6\$ ; ISOLATION SUGGEST RUNNING MEMORY MGMT. DIAG.  
3\$: NOP ; PSW=KERNEL MODE, PREV USER MODE  
MFPI SP ; GET PREV. MODE'S STACK POINTER  
MFPI @ (SP)+ ; GET DATA (AN ADDRESS) ON PREV MODE'S STACK  
MFPI @ (SP) ; GET DATA (=0) FROM PREV MODE'S ADDRESS  
NOP ; SPACE AND PUSH ONTO KERNEL STACK  
BNE 4\$ ; ERROR IF BRANCH TAKEN! SHOULD HAVE A ZERO ON THE STACK

```

7286 031346 005116          COM      (SP)          ;COMPLEMENT OPERAND
7287 031350 006636          MTPI     @ (SP)+        ;POP OPERAND OFF KERNEL STACK AND MOVE
7288                                ;IT TO PREV MODE'S SPACE
7289 031352 000002          RTI                      ;RETURN TO INST FOLLOWING IOT ABOVE
7290 031354 104000          5$:    HLT                      ;ERROR! MEMORY MANG. ABORT
7291 031356 105037 177776          CLRB     @#PSW          ;SET PRIORITY LEVEL BACK TO 0
7292 031362 012737 054662 000250 6$:    MOV     @KTABRT,@#MVEC ;RESTORE VECTOR
7293 031370 012737 046720 000020          MOV     @SSCOPE,@#IOTVEC
7294 031376 012706 000700          MOV     @USESTK,SP     ;RESTORE STACK POINTER
7295                                ;*****
7296                                ;*TEST 75      CHECK ILLEGAL HALT
7297                                ;*****
7298 031402                                †ST75:
7299 031402 000004          SCOPE
7300 031404 112737 000075 001202          MOV     @75,@#STSTNM
7301 031412 010746          HALT1:  MOV     PC,-(SP)      ;GET CURRENT PC
7302 031414 062716 000022          ADD     @2$-.,(SP)
7303 031420 011637 000004          MOV     (SP),@#ERRVEC  ;SET ERROR TRAP VECTOR TO 2$ BELOW
7304 031424 012637 000010          MOV     (SP)+,@#RESVEC ;LOAD RESERVED INST TRAP VECTOR
7305 031430 000000          HALT
7306 031432 104000          1$:    HLT                      ;SHOULD TRAP TO 4 IN USER MODE
7307 031434 000404          BR      3$
7308 031436 010716          2$:    MOV     PC,(SP)      ;REPLACE RETURN PC WITH
7309 031440 062716 000006          ADD     @3$-.,(SP)     ;ADDRESS OF 3$ BELOW
7310 031444 000002          RTI                      ;RETURN (TO 3$)
7311
7312 031446 012737 055064 000004 3$:    MOV     @ERPRT,@#ERRVEC ;RESTORE ERROR TRAP VECTOR
7313 031454 012737 054774 000010          MOV     @RESERR,@#RESVEC
7314 031462 105037 177776          CLRB     @#PSW
7315                                ;*****
7316                                ;*TEST 76      CHECK RESET IN USER MODE
7317                                ;*****
7318 031466                                †ST76:
7319 031466 000004          SCOPE
7320 031470 112737 000076 001202          MOV     @76,@#STSTNM
7321 031476 000277          RESET1: SCC
7322 031500 013700 177776          MOV     @#PSW,R0      ;GET CURRENT PSW
7323 031504 000277          SCC
7324 031506 000005          RESET
7325 031510 023700 177776          CMP     @#PSW,R0      ;CHECK THAT PSW UNCHANGED BY RESET ABOVE
7326 031514 001401          BEQ     .+4
7327 031516 104000          HLT                      ;ERROR! RESET CLEARED MODE BITS IN PSW
7328 031520 010037 177776          MOV     R0,@#PSW     ;RESTORE PSW (FOR ERROR)
7329 031524          ENDCP:
7330 031524 000004          RELE6:  SCOPE
7331 031526 010702          MOV     PC,R2
7332 031530 062702 000012          ADD     @12,R2
7333 031534 012707 036574          MOV     @RELOC,PC     ;GO RELOCATE PROGRAM CODE
7334 031540 000000          REL66:  .WORD 0
7335                                ;*****
7336                                ;*****
7337                                ;*****
7338                                ;*TEST 77      TEST STACK LIMIT REGISTER
7339                                ;*****
7340 031542                                †ST77:
7341 031542 012767 000001 147554          MOV     @1,STIMES     ;;DO 1 ITERATION

```





```

7398 031754 005066 000336 20$: CLR 336(SP) ; SHOULD CAUSE 'RED ZONE' TRAP
7399 031760 012706 000700 3$: MOV #USESTK, SP ; RESTORE THE STACK
7400 031764 104000 HLT ; ERROR! FAILED TO TRAP
7401
7402 031766 032737 140000 000002 4$: BIT #UM, @#2 ; CHECK IF TRAPPED WHEN IN USER
7403 ; MODE (2 CONTAINS OLD PSW)
7404 031774 001013 BNE 99$ ; GO TO ERROR CALL
7405 031776 010600 MOV SP, R0 ; STACK PTR SHOULD = 0
7406 032000 001011 BNE 99$ ; GO TO ERROR CALL IF NOT 0
7407 032002 026304 000336 CMP 336(R3), R4 ; CHECK THAT INST WAS ABORTED
7408 032006 001006 BNE 99$ ; GO REPORT ERRPR
7409 032010 005012 5$: CLR (R2) ; CLEAR STACK LIMIT REG
7410 032012 010705 MOV PC, R5 ; GET CURRENT PC
7411 032014 062705 177744 ADD #3$, R5 ; FORM ADDRESS OF 3$ ABOVE
7412 032020 020516 CMP R5, (SP) ; CHECK THAT RETURN PC IS ON
7413 ; THE STACK (AT 0)
7414 032022 001406 BEQ 100$ ; EXIT TEST
7415
7416 ; ERROR
7417 032024 005012 99$: CLR (R2) ; CLEAR STACK LIMIT REG
7418 032026 010463 000336 MOV R4, 336(R3) ; RESTORE MEM LOCATION
7419 032032 012706 000700 MOV #USESTK, SP ; SET STACK PTR
7420 032036 104000 HLT ; ERROR!
7421 032040 010463 000336 100$: MOV R4, 336(R3) ; RESTORE MEM LOCATION
7422 032044 005022 CLR (R2)+ ; CLEAR STACK LIM REG
7423 032046 012706 000700 MOV #USESTK, SP ; SET STACK PTR
7424 032052 042712 000340 BIC #340, (R2) ; SET PRIORITY LEVEL BACK TO 0
7425 032056 012737 055064 000004 MOV #ERPRT, @#ERRVEC ; RESTORE ERROR TRAP VECTOR
7426 032064 013737 177776 000006 MOV @#PSW, @#ERRVEC+2
7427 032072 112737 000340 000006 MOVB #PR7, @#ERRVEC+2
7428 032100 101$:
7429 ; *****
7430 ; *TEST 100 MEMORY MANAGEMENT REGISTER TESTS
7431 ; * PDR TEST - THIS TEST WRITES 64. RANDOM #'S INTO EACH PDR REGISTER
7432 ; * NOTE: IF MEM MGMT IS ENABLED ONLY PDR/PAR PAIRS 3-5 ARE TESTED.
7433 ; *****
7434 032100 †ST100:
7435 032100 000004 SCOPE
7436 032102 112737 000100 001202 MOVB #100, @#STSTNM
7437
7438 032110 012702 032340 KTPDR: MOV #PDRtbl, R2 ; SET TABLE ADDRESS OF PDR'S
7439 032114 012705 100361 MOV #100361, R5 ; SET BIT MASK
7440 032120 012200 1$: MOV (R2)+, R0 ; GET PDR ADDRESS
7441 032122 001435 BEQ 100$ ; EXIT ON '0' TERMINATOR
7442 032124 012716 000010 2$: MOV #8, (SP) ; SET LOOP COUNT (FOR 8 REGS)
7443 032130 105737 001545 TSTB @#MMON ; BRANCH IF MEM MGMT DISABLED
7444 032134 001404 BEQ 3$
7445 032136 062700 000006 ADD #6, R0 ; SET R0 TO PDR3
7446 032142 012716 000003 MOV #3, (SP) ; AND LIMIT TO TEST 3 PDRS
7447 032146 012703 000040 3$: MOV #32., R3 ; SET DATA COUNT
7448 032152 005004 CLR R4 ; INITIALIZE DATA TO BE WRITTEN
7449 032154 040504 4$: BIC R5, R4 ; CLEAR NON-SETTABLE BITS
7450 032156 010410 MOV R4, (R0) ; WRITE INTO PDR
7451 032160 021004 CMP (R0), R4 ; AND CHECK DATA READ BACK
7452 032162 001013 BNE 99$ ; GO TO ERROR CALL
7453 032164 005104 COM R4 ; COMPLEMENT DATA

```



```

7454 032166 040504          BIC    R5,R4          ;CLEAR NON-SETTABLE BITS
7455 032170 010410          MOV    R4,(R0)        ;WRITE COMPLEMENT DATA INTO PDR
7456 032172 021004          CMP    (R0),R4        ;AND CHECK
7457 032174 001006          BNE   99$             ;GO TO ERROR CALL
7458 032176 060104          ADD    R1,R4          ;STEP DATA
7459 032200 077313          SOB   R3,4$          ;
7460 032202 005020          5$:   CLR    (R0)+      ;STEP TO NEXT REGISTER
7461 032204 005316          DEC    (SP)           ;DECREMENT REGISTER COUNT
7462 032206 001357          BNE   3$              ;
7463 032210 000743          BR    1$              ;GET NEXT SET OF 8 REGISTERS
7464
7465 032212 104000          99$:  HLT              ;ERROR! INCORRECT DATA READ
7466                                     ;BACK FROM PDR. ADDRESS OF
7467                                     ;PDR IS IN R0, DATA IS IN R4
7468 032214 000772          BR    5$              ;STEP TO NEXT REGISTER
7469 032216
7470                                     ;*****
7471                                     ;*TEST 101 PAR TEST
7472                                     ;* PAR TEST - THIS TEST WRITES 64. COMPLEMENTING RANDOM #'S INTO EACH PAR.
7473                                     ;*****
7474 032216
7475 032216 000004          TST101: SCOPE
7476 032220 112737 000101 001202  MOVB   #101,2#STSTNM
7477 032226 012702 032346  KTPAR: MOV    #PARTBL,R2 ;GET TABLE ADDRESS OF PAR'S
7478 032232 005005          CLR    R5
7479 032234 012705 170000  MOV    #170000,R5
7480 032240 012200          1$:   MOV    (R2)+,R0 ;GET PAR ADDRESS
7481 032242 001435          BEQ   100$            ;EXIT ON '0' TERMINATOR
7482 032244 012716 000010  2$:   MOV    #8,(SP) ;SET LOOP COUNT (FOR 8 REGS.)
7483 032250 105737 001545  TSTB  2#MMON ;BRANCH IF MEM MGMT DISABLED
7484 032254 001404          BEQ   3$
7485 032256 062700 000006  ADD    #6,R0 ;SET R0 TO PAR3
7486 032262 012716 000003  MOV    #3,(SP) ;AND LIMIT TEST TO 3 PARS
7487 032266 012703 000040  3$:   MOV    #32.,R3 ;SET DATA COUNT
7488 032272 005004          CLR    R4 ;INITIALIZE DATA
7489 032274 040504          4$:   BIC    R5,R4 ;CLEAR NON-SETTABLE BITS
7490 032276 010410          MOV    R4,(R0) ;WRITE INTO PAR
7491 032300 021004          CMP    (R0),R4 ;AND CHECK
7492 032302 001013          BNE   99$            ;TAKE ERROR EXIT
7493 032304 005104          COM   R4 ;COMPLEMENT DATA
7494 032306 040504          BIC    R5,R4 ;CLEAR NON-SETTABLE BITS
7495 032310 010410          MOV    R4,(R0) ;WRITE COMPLEMENT DATA
7496 032312 021004          CMP    (R0),R4 ;AND CHECK
7497 032314 001006          BNE   99$            ;TAKE ERROR EXIT
7498 032316 060104          ADD    R1,R4 ;STEP DATA
7499 032320 077313          SOB   R3,4$          ;LOOP UNTIL FINISHED
7500
7501 032322 005020          5$:   CLR    (R0)+      ;DECREMENT REGISTER COUNT
7502 032324 005316          DEC    (SP)           ;BRANCH IF 8 REGS NOT DONE
7503 032326 001357          BNE   3$
7504 032330 000743          BR    1$
7505
7506 032332 104000          99$:  HLT              ;ERROR! INCORRECT DATA READ BACK
7507                                     ;FROM PAR. ADDRESS OF PAR IS IN
7508                                     ;R0, DATA IS IN R4
7509 032334 000772          BR    5$              ;DO NEXT REGISTER

```

7510 032336  
 7511 032336 000406  
 7512  
 7513 032340 172300  
 7514 032342 177600  
 7515 032344 000000  
 7516  
 7517 032346 172340  
 7518 032350 177640  
 7519 032352 000000  
 7520  
 7521  
 7522  
 7523  
 7524  
 7525  
 7526  
 7527 032354  
 7528 032354 000004  
 7529 032356 112737 000102 001202  
 7530 032364 105737 001545  
 7531 032370 001477  
 7532 032372 005037 172350  
 7533 032376 005037 172310  
 7534 032402 005037 177650  
 7535 032406 005037 177610  
 7536 032412 013746 000250  
 7537 032416 013746 000252  
 7538 032422 010746  
 7539 032424 062716 000040  
 7540 032430 012637 000250  
 7541 032434 013737 177776 000252  
 7542 032442 005000  
 7543 032444 010702  
 7544 032446 012703 100000  
 7545 032452 014223  
 7546 032454 005700  
 7547 032456 001001  
 7548 032460 104000  
 7549 032462 000433  
 7550  
 7551 032464 013700 177776  
 7552 032470 000300  
 7553 032472 006200  
 7554 032474 042700 177637  
 7555 032500 062700 100011  
 7556 032504 020037 177572  
 7557 032510 001013  
 7558 032512 012700 032452  
 7559 032516 020037 177576  
 7560 032522 001006  
 7561 032524 012700 032452  
 7562 032530 005720  
 7563 032532 020016  
 7564 032534 001001  
 7565 032536 000002

```

100$:
BR TST102 ;GO TO NEXT TEST
;TABLES FOR PDR & PAR TESTS ABOVE
PDRtbl: .WORD KIPDR0
        .WORD UIPDR0
        .WORD 0 ;TERMINATOR

PARTBL: .WORD KIPAR0
        .WORD UIPAR0
        .WORD 0 ;TERMINATOR

;*****
;TEST 102 CHECK KT ABORT LOGIC
; THIS TEST CHECKS KT ABORT LOGIC. TEST CREATES AN ABORT CONDITION
; AND INSURES THAT ABORT IS TAKEN PROPERLY. NOTE: TEST IS EXECUTED ONLY
; IF TEST IS ENTERED WITH MEM MGMT ENABLED.
;*****
TST102:
SCOPE
MOVB #102, @#STSTNM
KTABT: TSTB @#MMON ;BRANCH IF MEM MGMT DISABLED
        BEQ KTEX
        CLR @#KIPAR4 ;SET UP MEM MGMT REGISTERS
        CLR @#KIPDR4 ;TO ABORT IF A MEMORY
        CLR @#UIPAR4 ;REFERENCE IS MADE TO
        CLR @#UIPDR4 ;ADDRESSES (VIRTUAL) BETWEEN
1$: MOV @#MMVEC, -(SP) ;SAVE MEM MGMT VECTOR
    MOV @#MMVEC+2, -(SP) ;AND PRIORITY
    MOV PC, -(SP) ;SET MEM MGMT
    ADD #4, -(SP) ;VECTOR TO 4$ BELOW
    MOV (SP)+, @#MMVEC
    MOV @#PSW, @#MMVEC+2
    CLR R0 ;CLEAR ABORT INDICATOR
    MOV PC, R2 ;SET R2 AND R3 NOTE:
    MOV #100000, R3 ;THE REF VIA R3 CAUSES THE
2$: MOV -(R2), (R3)+ ;ABORT
3$: TST R0 ;BRANCH IF THE ABORT OCCURRED
    BNE .+4
    HLT ;REPORT ERROR
    BR 100$

;ABORT HERE
4$: MOV @#PSW, R0 ;SR0 SHOULD CONTAIN
    SWAB R0 ;CAUSE FOR ABORT AND
    ASR R0 ;ALSO WHICH SEGMENT
    BIC #177637, R0 ;WAS IN USE WHEN ABORT
    ADD #100011, R0 ;OCCURRED.
    CMP R0, @#SR0
    BNE 99$
    MOV #2$, R0 ;GET ADDRESS OF INST THAT ABORTED
    CMP R0, @#SR2 ;THAT ABORTED
    BNE 99$
    MOV #2$, R0
5$: TST (R0)+ ;R0=ADDRESS OF INST FOLLOWING ABORT
    CMP R0, (SP) ;(3$)
    BNE 99$
    RTI ;RETURN
    
```



```

7566          :ENTER HERE ON ERROR
7567 032540 104000 99$: HLT ;REPORT ERROR
7568 032542 010716  MOV PC,(SP)
7569 032544 062716 177710 ADD #3$-.,(SP)
7570 032550 000002  RTI ;RETURN
7571 032552 012637 000252 100$: MOV (SP)+,2#MMVEC+2 ;RESTORE ABORT VECTOR
7572 032556 012637 000250  MOV (SP)+,2#MMVEC ;& PRIORITY.
7573 032562 012737 000001 177572  MOV #1,2#SRO ;CLEAR ERROR CONDITIONS
7574 032570
7575 032570 000004  KTEX:
7576 032572 010702  RELE7: SCOPE
7577 032574 062702 000012  MOV PC,R2
7578 032600 012707 036574  ADD #12,R2
7579 032604 000000  MOV #RELOC,PC ;GO RELOCATE PROGRAM CODE
7580  REL77: .WORD 0
7581  ;7777777777777777 LAST ADDRESS OF CODE TO BE RELOCATED 777777777777
7582  ;*****
7583  ;*TEST 103 FLOATING POINT TEST 1
7584  ;*
7585  ;* THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
7586  ;* COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
7587  ;* EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
7588  ;* SECT1 (A+B)**2=A**2+2*A*B+B**2
7589  ;* SECT2 (A+B)*(A-B)=A**2-B**2
7590  ;* SECT3 A/B*B=A
7591  ;*****
7592 032606 146510 †ST103: MOV #1,$TIMES ;;DO 1 ITERATION
7593 032606 012767 000001  SCOPE
7594 032614 000004  MOVB #103,2#$STSTNM
7595 032616 112737 000103 001202  .SBTTL START OF SECTION B
7596  ;8888888888888888 FIRST ADDRESS TO BE RELOCATED 8888888888
7597 7598 032624 112737 000103 001202 RELB: MOVB #STN-1,2#$STSTNM
7599 032632 010700  MOV PC,RO ;GET PC
7600 032634 005740  TST -(RO) ;RO CONTAINS THE ADDRESS OF RELB
7601 032636 010037 001554  MOV RO,2#FRSTAD ;SAVE
7602 032642 010700  MOV PC,RO ;GET CURRENT PC
7603 032644 162700 032644  SUB #,RO ;SUBTRACT RELOCATION FACTOR
7604 032650 010037 001550  MOV RO,2#FACTOR ;SAVE RELOCATION FACTOR
7605 032654 010737 001212  MOV PC,2#$LPERR ;SET LOOP ADDRESS
7606 032660 062737 000026 001212  ADD #26,2#$LPERR ;ADJUST
7607 032666 013737 001212 001210  MOV 2#$LPERR,2#$LPADR
7608 032674 105737 001544  TSTB 2#NEXEC ;BR IF TEST CODE TO BE EXECUTED
7609 032700 001402  BEQ .+6
7610 032702 000167 002400  JMP RELEB
7611 032706 004737 053056 100$: JSR PC,2#FLTSGL ;GET RANDOM OPERANDS
7612 032712 170127 000000  LDFPS #0 ;INIT FPS
7613 032716 172537 001424  LDF 2#FTMP0,AC1 ;LOAD A OPERAND
7614 032722 172437 001430  LDF 2#FTMP2,AC0 ;LOAD B OPERAND
7615 032726 013737 001444 001412  MOV 2#FREG0,2#$AC1 ;SETUP EXTENDED
7616 032734 013737 001446 001410  MOV 2#FREG1,2#$AC0 ;EXPONENTS
7617 032742 004767 002202  JSR PC,FLTADD ;PERFORM THE ADD
7618 032746 174100  STF AC1,AC0 ;SETUP AC0 TO
7619 032750 013737 001412 001410  MOV 2#$AC1,2#$AC0 ;PERFORM THE SQUARE
7620 032756 004767 002116  JSR PC,FLTMPY ;DO THE MULTIPLY

```

7622	032762	174137	001434		STF	AC1, @FTMP4	;SAVE RESULT
7623	032766	013737	001412	001450	MOV	@SAC1, @FREG2	;AND SOFTWARE EXP
7624							
7625							
7626							
7627	032774	013737	001444	001410			
7628	033002	172437	001424				
7629	033006	013737	001410	001412			
7630	033014	172500					
7631	033016	004767	002056				
7632	033022	174102					
7633	033024	013737	001412	001414			
7634							
7635							
7636	033032	172437	001430				
7637	033036	172500					
7638	033040	013737	001446	001410			
7639	033046	013737	001410	001412			
7640	033054	004767	002020				
7641	033060	174103					
7642	033062	013737	001412	001416			
7643							
7644							
7645	033070	012701	001430				
7646	033074	172411					
7647	033076	172541					
7648	033100	013737	001446	001410			
7649	033106	013737	001444	001412			
7650	033114	004767	001760				
7651	033120	172427	040000				
7652	033124	012737	000002	001410			
7653	033132	004767	001742				
7654							
7655							
7656	033136	013737	001416	001410			
7657	033144	172403					
7658	033146	004767	001776				
7659	033152	172402					
7660	033154	013737	001414	001410			
7661	033162	004767	001762				
7662	033166	174137	001440				
7663	033172	013737	001412	001452			
7664							
7665							
7666							
7667							
7668	033200	023737	001414	001416			
7669	033206	002003					
7670	033210	013737	001416	001414			
7671	033216	163737	001412	001414			
7672	033224	162737	000024	001414			
7673	033232	005437	001414				
7674	033236	172437	001434				
7675	033242	013737	001450	001410			
7676	033250	004767	001670				
7677	033254	163737	001452	001412			

;NOW DO THE RIGHT HAND SIDE OF THE EQUATION  
 ;DO THE A\*A FIRST  
 ;NOW DO THE B\*B  
 ;NOW DO THE 2\*B\*A  
 ;NOW SUM THE RESULTS  
 ;NOW CHECK BOTH SIDES OF THE EQUATION  
 ;CALCULATE THE NUMBER OF CORRECT BITS  
 ;PUT LARGEST EXPONENT OF A\*\*2 OR B\*\*2 IN SAC2  
 ;BRANCH IF SAC2 ALREADY HAS LARGEST  
 ;SAC3 WAS LARGER  
 ;NOW CALCULATE NUMBER  
 ;OF CORRECT BITS WITHIN 2  
 ;MAKE RESULT POSITIVE  
 ;LOAD RESULT OF LEFT HAND SIDE  
 ;AND EXTENDED EXPONENT  
 ;SUBTRACT TO SEE HOW CLOSE THEY ARE  
 ;GET DIFFERENCE IN EXT EXPONENTS



```

7678                                     ;ACTUAL EXP'S ARE EQUAL TO 200
7679 033262 100002 BPL 35 ;ENSURE RESULT IS POSITIVE
7680 033264 005437 001412 NEG 2#SAC1
7681 033270 023737 001414 001412 3S: CMP 2#SAC2,2#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
7682 033276 003401 BLE SECT2 ;BRANCH IF YES
7683 033300 104014 4S: ERROR 14 ;RESULTS ARE WRONG
7684 ..*****
7685 033302 170127 000000 SECT2: LDFPS #0
7686 ;DO A+B
7687 033306 172537 001424 LDF 2#FTMP0,AC1 ;LOAD A OPERAND
7688 033312 172437 001430 LDF 2#FTMP2,AC0 ;LOAD B OPERAND
7689 033316 013737 001444 001412 MOV 2#FREG0,2#SAC1
7690 033324 013737 001446 001410 MOV 2#FREG1,2#SAC0
7691 033332 004767 001612 JSR PC,FLTADD ;ADD THEM
7692 033336 174102 STF AC1,AC2 ;SAVE IN AC2
7693 033340 013737 001412 001414 MOV 2#SAC1,2#SAC2 ;AND EXT EXPONENT
7694 ;NOW DO THE A-B
7695 033346 172537 001424 LDF 2#FTMP0,AC1 ;LOAD OPERAND A
7696 033352 013737 001444 001412 MOV 2#FREG0,2#SAC1 ;AND EXT EXPONENT
7697 033360 172437 001430 LDF 2#FTMP2,AC0 ;LOAD OPERAND B
7698 033364 013737 001446 001410 MOV 2#FREG1,2#SAC0
7699 033372 004767 001546 JSR PC,FLTSUB ;SUBTRACT THEM
7700 ;NOW DO (A+B)*(A-B)
7701 033376 172402 LDF AC2,AC0 ;GET RESULT OF (A+B)
7702 033400 013737 001414 001410 MOV 2#SAC2,2#SAC0
7703 033406 004767 001466 JSR PC,FLTMPY ;FORM THE PRODUCT
7704 033412 174137 001434 STF AC1,2#FTMP4 ;SAVE RESULT
7705 033416 013737 001412 001450 MOV 2#SAC1,2#FREG2 ;AND EXT EXPONENT
7706 ;NOW DO THE B*B
7707 033424 172437 001430 LDF 2#FTMP2,AC0 ;LOAD OPERAND B
7708 033430 013737 001446 001410 MOV 2#FREG1,2#SAC0
7709 033436 172500 LDF AC0,AC1 ;B OPERAND IS IN AC0
7710 033440 013737 001410 001412 MOV 2#SAC0,2#SAC1 ;AND EXT EXPONENT
7711 033446 004767 001426 JSR PC,FLTMPY
7712 033452 174102 STF AC1,AC2 ;SAVE RESULT IN AC2
7713 033454 013737 001412 001414 MOV 2#SAC1,2#SAC2
7714 ;NOW DO THE A*A
7715 033462 172437 001424 LDF 2#FTMP0,AC0 ;LOAD OPERAND A
7716 033466 013737 001444 001410 MOV 2#FREG0,2#SAC0
7717 033474 172500 LDF AC0,AC1
7718 033476 013737 001410 001412 MOV 2#SAC0,2#SAC1
7719 033504 004767 001370 JSR PC,FLTMPY ;EXECUTE THE MULTIPLY
7720 033510 013737 001412 001416 MOV 2#SAC1,2#SAC3 ;SAVE EXT EXPO OF A*A
7721 ;NOW DO A**2-B**2
7722 033516 172402 LDF AC2,AC0 ;GET B*B
7723 033520 013737 001414 001410 MOV 2#SAC2,2#SAC0 ;A*A IN AC1
7724 033526 004767 001412 JSR PC,FLTSUB
7725 033532 174137 001440 STF AC1,2#FTMP6 ;SAVE IN MEMORY
7726 033536 013737 001412 001452 MOV 2#SAC1,2#FREG3
7727 ;NOW COMPUTE THE RESULTS
7728 ;CALCULATE THE NUMBER OF CORRECT BITS
7729 033544 023737 001414 001416 CMP 2#SAC2,2#SAC3 ;DETERMINE WHICH EXP IS LARGER
7730 033552 002003 BGE 25 ;BRANCH IF AC2 LARGER
7731 033554 013737 001416 001414 MOV 2#SAC3,2#SAC2 ;PUT LARGEST IN AC2
7732 033562 163737 001412 001414 2S: SUB 2#SAC1,2#SAC2
7733 033570 162737 000025 001414 SUB #21.,2#SAC2

```

```

7734 033576 005437 001414      NEG      2#SAC2
7735 033602 172437 001434      LDF      2#FTMP4,AC0      ;GET LEFT HAND SIDE
7736 033606 013737 001450 001410      MOV      2#FREG2,2#SAC0
7737 033614 004767 001324      JSR      PC,FLTSUB      ;SUBTRACT TO SEE HOW CLOSE THEY ARE
7738 033620 163737 001452 001412      SUB      2#FREG3,2#SAC1  ;SUB EXT EXPONENTS
7739                                     ;ACTUAL EXPONENTS ARE EQUAL
7740 033626 100002                                     ;MAKE SURE RESULT IS POSITIVE
7741 033630 005437 001412      BPL      15
7742 033634 023737 001414 001412 15:      NEG      2#SAC1
7743 033642 003401                                     ;RESULTS WITHIN RANGE ALLOWED?
7744 033644 104014      CMP      2#SAC2,2#SAC1  ;BRANCH IF YES
7745                                     BLE      SECT3
7746                                     ERROR    14      ;RESULTS WRONG

```

```

7747 033646 172537 001424      ;*****
7748 033652 172437 001430      SECT3:  LDF      2#FTMP0,AC1  ;LOAD OPERAND A
7749 033656 013737 001444 001412      LDF      2#FTMP2,AC0      ;AND OPERAND B
7750 033664 013737 001446 001410      MOV      2#FREG0,2#SAC1
7751 033672 004767 001224      MOV      2#FREG1,2#SAC0
7752 033676 004767 001176      JSR      PC,FLTDIV      ;GO DIVIDE THEM
7753 033702 174137 001434      JSR      PC,FLTMPY      ;MULTIPLY RESULT BY B
7754 033706 013737 001412 001450      STF      AC1,2#FTMP4      ;SAVE RESULT
7755 033714 172437 001424      MOV      2#SAC1,2#FREG2
7756 033720 174037 001440      LDF      2#FTMP0,AC0      ;LOAD OPERAND A
7757 033724 013737 001444 001410      STF      AC0,2#FTMP6      ;SAVE INCASE TYPE OUT
7758 033732 013737 001444 001452      MOV      2#FREG0,2#SAC0
7759 033740 004767 001200      MOV      2#FREG0,2#FREG3
7760 033744 163737 001444 001412      JSR      PC,FLTSUB      ;SUBTRACT RIGHT AND LEFT HAND SIDES
7761 033752 100002      SUB      2#FREG0,2#SAC1  ;SEE IF RESULT OK
7762 033754 005437 001412      BPL      15      ;ENSURE DIFFERENCE IS POSITIVE
7763 033760 022737 000027 001412 15:      NEG      2#SAC1
7764 033766 003001      CMP      23.,2#SAC1      ;RESULTS WITHIN 2 BITS?
7765 033770 000401      BGT      25      ;BRANCH IF NO
7766 033772 104014      BR      TST104      ;GO TO NEXT TEST
7767                                     ;RESULTS WRCNG
7768                                     ;*****

```

```

7769 *TEST 104      FLOATING POINT TEST 2
7770 *
7771 *      THIS TEST TAKES TWO RANDOM NUMBERS (A AND B) AND
7772 *      COMPARES THE RESULTS OF TWO EQUAL CALCULATIONS.
7773 *      EACH SECTION EVALUATES A DIFFERENT EQUATION AS DESCRIBED BELOW:
7774 *      SECT1      (A+B)**2=A**2+2*A*B+B**2
7775 *      SECT2      (A+B)*(A-B)=A**2-B**2
7776 *      SECT3      A/B*B=A
7777 ;*****

```

```

7778 †TST104:
7779 033774 000004      SCOPE
7780 033776 112737 000104 001202      MOV      #104,2#STSTNM
7781 034004 012737 000001 001324      MOV      #1,2#STIMES
7782 034012 004737 053050 1005:      JSR      PC,2#FLTDBL      ;GET RANDOM OPERANDS
7783 034016 170127 000200      LDFPS    #200      ;INIT FPS
7784 034022 172537 001424      LDF      2#FTMP0,AC1      ;LOAD A OPERAND
7785 034026 172437 001434      LDF      2#FTMP4,AC0      ;LOAD B OPERAND
7786 034032 013737 001444 001412      MOV      2#FREG0,2#SAC1  ;SETUP EXTENDED
7787 034040 013737 001446 001410      MOV      2#FREG1,2#SAC0  ;EXONENTS
7788 034046 004767 001076      JSR      PC,FLTADD      ;PERFORM THE ADD
7789 034052 174100      STF      AC1,AC0      ;SETUP AC0 TO

```



```

7790 034054 013737 001412 001410 MOV 2#SAC1,2#SAC0 ;PERFORM THE SQUARE
7791 034062 004767 001012 JSR PC,FLTMPY ;DO THE MULTIPLY
7792 034066 174137 001464 STF AC1,2#FLTMP0 ;SAVE RESULT
7793 034072 013737 001412 001450 MOV 2#SAC1,2#FREG2 ;AND SOFTWARE EXP
7794
7795 ;NOW DO THE RIGHT HAND SIDE OF THE EQUATION
7796 ;DO THE A*A FIRST
7797 034100 013737 001444 001410 MOV 2#FREG0,2#SAC0 ;GET EXT EXPONENT
7798 034106 172437 001424 LDF 2#FTMP0,AC0 ;LOAD OPERAND A
7799 034112 013737 001410 001412 MOV 2#SAC0,2#SAC1 ;SET OPERAND B EXT EXPONENT
7800 034120 172500 LDF AC0,AC1 ;LOAD B OPERAND
7801 034122 004767 000752 JSR PC,FLTMPY ;EXECUTE THE MULTIPLY
7802 034126 174102 STF AC1,AC2 ;SAVE RESULT
7803 034130 013737 001412 001414 MOV 2#SAC1,2#SAC2
7804
7805 ;NOW DO THE B*B
7806 034136 172437 001434 LDF 2#FTMP4,AC0 ;LOAD B OPERAND
7807 034142 172500 LDF AC0,AC1
7808 034144 013737 001446 001410 MOV 2#FREG1,2#SAC0 ;AND EXT EXPONENT
7809 034152 013737 001410 001412 MOV 2#SAC0,2#SAC1
7810 034160 004767 000714 JSR PC,FLTMPY ;DO THE MULTIPLY
7811 034164 174103 STF AC1,AC3 ;SAVE THE RESULT
7812 034166 013737 001412 001416 MOV 2#SAC1,2#SAC3
7813
7814 ;NOW DO THE 2*B*A
7815 034174 012701 001434 MOV 2#FTMP4,R1
7816 034200 172411 LDF (R1),AC0 ;LOAD THE B OPERAND
7817 034202 172541 LDF -(R1),AC1 ;LOAD THE A OPERAND
7818 034204 013737 001446 001410 MOV 2#FREG1,2#SAC0 ;AND THE EXT EXPONENTS
7819 034212 013737 001444 001412 MOV 2#FREG0,2#SAC1
7820 034220 004767 000654 JSR PC,FLTMPY ;DO THE MULTIPLY
7821 034224 172427 040000 LDF 2#040000,AC0 ;SETUP TO MULTIPLY BY TWO
7822 034230 012737 000002 001410 MOV 2#2,2#SAC0
7823 034236 004767 000636 JSR PC,FLTMPY ;DO THE MULTIPLY
7824
7825 ;NOW SUM THE RESULTS
7826 034242 013737 001416 001410 MOV 2#SAC3,2#SAC0
7827 034250 172403 LDF AC3,AC0 ;GET RESULT OF B*B
7828 034252 004767 000672 JSR PC,FLTADD ;ADD THE RESULT
7829 034256 172402 LDF AC2,AC0 ;GET RESULT OF A*A
7830 034260 013737 001414 001410 MOV 2#SAC2,2#SAC0
7831 034266 004767 000656 JSR PC,FLTADD ;ADD THIS RESULT
7832 034272 174137 001474 STF AC1,2#FLTMP1 ;SAVE FINAL RESULT
7833 034276 013737 001412 001452 MOV 2#SAC1,2#FREG3
7834
7835 ;NOW CHECK BOTH SIDES OF THE EQUATION
7836 ;CALCULATE THE NUMBER OF CORRECT BITS
7837 ;PUT LARGEST EXPONENT OF A**2 OR B**2 IN SAC2
7838 034304 023737 001414 001416 CMP 2#SAC2,2#SAC3
7839 034312 002003 BGE 15 ;BRANCH IF SAC2 ALREADY HAS LARGEST
7840 034314 013737 001416 001414 MOV 2#SAC3,2#SAC2 ;SAC3 WAS LARGER
7841 034322 163737 001412 001414 15: SUB 2#SAC1,2#SAC2 ;NOW CALCULATE NUMBER
7842 034330 162737 000065 001414 SUB 2#53,2#SAC2 ;OF CORRECT BITS WITHIN 2
7843 034336 005437 001414 NEG 2#SAC2 ;MAKE RESULT POSITIVE
7844 034342 172437 001464 LDF 2#FLTMP0,AC0 ;LOAD RESULT OF LEFT HAND SIDE
7845 034346 013737 001450 001410 MOV 2#FREG2,2#SAC0 ;AND EXTENDED EXPONENT
    
```

```

7846 034354 004767 000564 JSR PC,FLTSUB ;SUBTRACT TO SEE HOW CLOSE THEY ARE
7847 034360 163737 001452 001412 SUB 2#FREG3,2#SAC1 ;GET DIFFERENCE IN EXT EXPONENTS
7848 ;ACTUAL EXP'S ARE EQUAL TO 200
7849 034366 100002 BPL 3S ;ENSURE RESULT IS POSITIVE
7850 034370 005437 001412 NEG 2#SAC1
7851 034374 023737 001414 001412 3S: CMP 2#SAC2,2#SAC1 ;ANSWERS WITHIN ALLOWABLE NUMBER?
7852 034402 003401 BLE SECT2D ;BRANCH IF YES
7853 034404 104011 4S: ERROR 11 ;RESULTS ARE WRONG
7854 ;*****
7855 034406 170127 000200 SECT2D: LDFPS #200
7856 ;DO A+B
7857 034412 172537 001424 LDF 2#FTMP0,AC1 ;LOAD A OPERAND
7858 034416 172437 001434 LDF 2#FTMP4,AC0 ;LOAD B OPERAND
7859 034422 013737 001444 001412 MOV 2#FREG0,2#SAC1
7860 034430 013737 001446 001410 MOV 2#FREG1,2#SAC0
7861 034436 004767 000506 JSR PC,FLTADD ;ADD THEM
7862 034442 174102 STF AC1,AC2 ;SAVE IN AC2
7863 034444 013737 001412 001414 MOV 2#SAC1,2#SAC2 ;AND EXT EXPONENT
7864 ;NOW DO THE A-B
7865 034452 172537 001424 LDF 2#FTMP0,AC1 ;LOAD OPERAND A
7866 034456 013737 001444 001412 MOV 2#FREG0,2#SAC1 ;AND EXT EXPONENT
7867 034464 172437 001434 LDF 2#FTMP4,AC0 ;LOAD OPERAND B
7868 034470 013737 001446 001410 MOV 2#FREG1,2#SAC0
7869 034476 004767 000442 JSR PC,FLTSUB ;SUBTRACT THEM
7870 ;NOW DO (A+B)*(A-B)
7871 034502 172402 LDF AC2,AC0 ;GET RESULT OF (A+B)
7872 034504 013737 001414 001410 MOV 2#SAC2,2#SAC0
7873 034512 004767 000362 JSR PC,FLTMPY ;FORM THE PRODUCT
7874 034516 174137 001464 STF AC1,2#FTMP0 ;SAVE RESULT
7875 034522 013737 001412 001450 MOV 2#SAC1,2#FREG2 ;AND EXT EXPONENT
7876 ;NOW DO THE B*B
7877 034530 172437 001434 LDF 2#FTMP4,AC0 ;LOAD OPERAND B
7878 034534 013737 001446 001410 MOV 2#FREG1,2#SAC0
7879 034542 172500 LDF AC0,AC1 ;B OPERAND IS IN AC0
7880 034544 013737 001410 001412 MOV 2#SAC0,2#SAC1 ;AND EXT EXPONENT
7881 034552 004767 000322 JSR PC,FLTMPY
7882 034556 174102 STF AC1,AC2 ;SAVE RESULT IN AC2
7883 034560 013737 001412 001414 MOV 2#SAC1,2#SAC2
7884 ;NOW DO THE A*A
7885 034566 172437 001424 LDF 2#FTMP0,AC0 ;LOAD OPERAND A
7886 034572 013737 001444 001410 MOV 2#FREG0,2#SAC0
7887 034600 172500 LDF AC0,AC1
7888 034602 013737 001410 001412 MOV 2#SAC0,2#SAC1
7889 034610 004767 000264 JSR PC,FLTMPY ;EXECUTE THE MULTIPLY
7890 034614 013737 001412 001416 MOV 2#SAC1,2#SAC3 ;SAVE EXT EXPO OF A*A
7891 ;NOW DO A**2-B**2
7892 034622 172402 LDF AC2,AC0 ;GET B*B
7893 034624 013737 001414 001410 MOV 2#SAC2,2#SAC0 ;A*A IN AC1
7894 034632 004767 000306 JSR PC,FLTSUB
7895 034636 174137 001474 STF AC1,2#FTMP1 ;SAVE IN MEMORY
7896 034642 013737 001412 001452 MOV 2#SAC1,2#FREG3
7897 ;NOW COMPUTE THE RESULTS
7898 ;CALCULATE THE NUMBER OF CORRECT BITS
7899 034650 023737 001414 001416 CMP 2#SAC2,2#SAC3 ;DETERMINE WHICH EXP IS LARGER

```



L13

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
DQKDCR.P11 07-FEB-77 09:58

MACY11 27(1006)  
T104 FLOATING POINT TEST 2

07-FEB-77 10:08 PAGE 167

7900 034656 002003

BGE 25

;BRANCH IF AC2 LARGER

```

7901 034660 013737 001416 001414      MOV      2#SAC3,2#SAC2      ;PUT LARGEST IN AC2
7902 034666 163737 001412 001414 2S:  SUB      2#SAC1,2#SAC2
7903 034674 162737 000066 001414      SUB      #54.,2#SAC2
7904 034702 005437 001414      NEG      2#SAC2
7905 034706 172437 001464      LDF      2#FLTMP0,AC0      ;GET LEFT HAND SIDE
7906 034712 013737 001450 001410      MOV      2#FREG2,2#SAC0
7907 034720 004767 000220      JSR      PC,FLTSUB
7908 034724 163737 001452 001412      SUB      2#FREG3,2#SAC1      ;SUBTRACT TO SEE HOW CLOSE THEY ARE
7909                                ;SUB EXT EXPONENTS
7910 034732 100002                                ;ACTUAL EXPONENTS ARE EQUAL
7911 034734 005437 001412                                ;MAKE SURE RESULT IS POSITIVE
7912 034740 023737 001414 001412 1S:  CMP      2#SAC2,2#SAC1
7913 034746 003401                                ;RESULTS WITHIN RANGE ALLOWED?
7914 034750 104011                                ;BRANCH IF YES
7915                                ;RESULTS WRONG
7916                                ;*****

```

```

7917 034752 172537 001424      SECT30: LDF      2#FTMP0,AC1      ;LOAD OPERAND A
7918 034756 172437 001434      LDF      2#FTMP4,AC0      ;AND OPERAND B
7919 034762 013737 001444 001412      MOV      2#FREG0,2#SAC1
7920 034770 013737 001446 001410      MOV      2#FREG1,2#SAC0
7921 034776 004767 000120      JSR      PC,FLTDIV      ;GO DIVIDE THEM
7922 035002 004767 000072      JSR      PC,FLTMPY      ;MULTIPLY RESULT BY B
7923 035006 174137 001464      STF      AC1,2#FLTMP0      ;SAVE RESULT
7924 035012 013737 001412 001450      MOV      2#SAC1,2#FREG2
7925 035020 172437 001424      LDF      2#FTMP0,AC0      ;LOAD OPERAND A
7926 035024 174037 001474      STF      AC0,2#FLTMP1      ;SAVE INCASE TYPE OUT
7927 035030 013737 001444 001410      MOV      2#FREG0,2#SAC0
7928 035036 013737 001444 001452      MOV      2#FREG0,2#FREG3
7929 035044 004767 000074      JSR      PC,FLTSUB      ;SUBTRACT RIGHT AND LEFT HAND SIDES
7930 035050 163737 001444 001412      SUB      2#FREG0,2#SAC1      ;SEE IF RESULT OK
7931 035056 100002                                ;ENSURE DIFFERENCE IS POSITIVE
7932 035060 005437 001412      BPL      1S
7933 035064 022737 000067 001412 1S:  CMP      #55.,2#SAC1
7934 035072 003505                                ;RESULTS WITHIN 2 BITS?
7935 035074 104011                                ;BRANCH IF YES
7936 035076 000503                                ;RESULTS WRONG
7937
7938                                ;*****

```

```

7939 .SBTTL  FLOATING POINT MULTIPLY ROUTINE
7940 .#      THIS ROUTINE MULTIPLIES THE CONTENTS OF AC0 AND AC1
7941 .#      AND LEAVES THE RESULT IN AC1. IT ALSO TAKES CARE OF
7942 .#      THE SOFTWARE EXPONENTS THAT ARE KEPT IN SAC0 AND SAC1.
7943 .#      *****
7944 035100 063737 001410 001412 FLTMPY: ADD      2#SAC0,2#SAC1      ;ADD SOFTWARE EXPONENTS
7945 035106 171100      MULF     AC0,AC1          ;DO THE MULTIPLY
7946 035110 012746 100400      MOV      #100400,-(SP)    ;PUT CONTROL WORD ON STACK
7947 035114 004737 053210      JSR      PC,2#EXPEXT      ;CALCULATE EXT EXPONENT
7948 035120 000207      RTS      PC              ;RETURN
7949
7950                                ;*****

```

```

7951 .SBTTL  FLOATING POINT DIVIDE ROUTINE
7952 .#      THIS ROUTINE DIVIDES THE CONTENTS OF AC1 BY AC0
7953 .#      AND LEAVES THE RESULT IN AC1.
7954 .#      *****
7955 035122 163737 001410 001412 FLTDIV: SUB      2#SAC0,2#SAC1      ;ADJUST SOFTWARE EXPONENTS
7956 035130 174500      DIVF     AC0,AC1          ;EXECUTE THE DIVIDE

```



7957 035132 012746 100400  
7958 035136 004737 053210  
7959 035142 000207  
7960  
7961  
7962  
7963  
7964  
7965  
7966  
7967  
7968 035144 010667 000134  
7969 035150 023737 001410 001412  
7970 035156 003016  
7971 035160 001434  
7972  
7973 035162 013702 001412  
7974 035166 163702 001410  
7975 035172 020227 000071  
7976 035176 002003  
7977  
7978 035200 005402  
7979 035202 176402  
7980 035204 000422  
7981 035206 176427 177703  
7982 035212 000417  
7983  
7984  
7985 035214 013702 001410  
7986 035220 163702 001412  
7987 035224 013737 001410 001412  
7988 035232 020227 000071  
7989 035236 002003  
7990 035240 005402  
7991 035242 176502  
7992 035244 000402  
7993  
7994  
7995 035246 176527 177703  
7996  
7997 035252 005767 000026  
7998 035256 001402  
7999 035260 173100  
8000 035262 000401  
8001 035264 172100  
8002 035266 012746 100400  
8003 035272 004737 053210  
8004 035276 005067 000002  
8005 035302 000207  
8006 035304 000000  
8007 035306 000004  
8008 035310 010702  
8009 035312 062702 000012  
8010 035316 012707 036574  
8011 035322 000000  
8012

```
MOV #100400, -(SP) ; PUT CONTROL WORD ON STACK
JSR PC, @EXPEXT ; CALCULATE EXT EXPONENT
RTS PC ; RETURN

;*****
;SBTTL FLOATING POINT ADD ROUTINE
; THIS ROUTINE ADDS THE CONTENTS OF ACO TO AC1.
; THIS CAN ONLY BE DONE IF THE SOFTWARE EXPONENTS
; ARE CLOSE ENOUGH TOGETHER SUCH THAT AN ADJUSTMENT
; OF THE REAL EXPONENT LEAVES A NON-ZERO NUMBER.
;*****
FLTSUB: MOV SP, SUBFLG ; SET SUBTRACT FLAG
FLTADD: CMP @SAC0, @SAC1 ; CHECK SOFTWARE EXPONENTS
        BGT 1$
        BEQ 2$
; ACCUMULATOR 1 IS LARGER THAN ACCUMULATOR 0
        MOV @SAC1, R2 ; GET OPERAND B SOFTWARE EXP
        SUB @SAC0, R2 ; GET DIFFERENCE IN SOFTWARE EXP'S
        CMP R2, #57. ; EXP WITHIN DBL PREC RANGE?
        BGE 7$ ; BRANCH IF ADD NOT REQUIRED
        ; RESULT IS OPERAND B
        NEG R2
        LDEXP R2, ACO ; RELOAD THE EXPONENT
        BR 2$
7$: LDEXP #-75, ACO ; FAKE EXPONENT SO HARDWARE
    BR 2$ ; WILL DETECT OUT OF RANGE

; ACCUMULATOR 0 IS LARGER THAN ACCUMULATOR 1
1$: MOV @SAC0, R2 ; GET SOFTWARE EXP OF OPERAND A
    SUB @SAC1, R2 ; GET DIFFERENCE IN EXP'S
    MOV @SAC0, @SAC1 ; MAKE SOFTWARE EXP'S EQUAL
    CMP R2, #57. ; EXP WITHIN DBL PREC RANGE?
    BGE 4$ ; BRANCH IF NO
    NEG R2
    LDEXP R2, AC1 ; RELOAD THE EXPONENT
    BR 2$

; ACCUMULATOR 0 IS MUCH LARGER THAN ACCUMULATOR 1 SO RESULT IS 0
4$: LDEXP #-75, AC1 ; FAKE EXPONENT SO HARDWARE
    ; WILL DETECT OUT OF RANGE
    TST SUBFLG ; ADD OR SUBTRACT?
    BEQ 5$ ; BRANCH IF ADD
    SUBF ACO, AC1
    BR 6$
5$: ADDF ACO, AC1 ; EXECUTE THE ADD
6$: MOV #100400, -(SP) ; PUT CONTROL WORD ON STACK
    JSR PC, @EXPEXT ; CALCULATE EXT EXPONENT
    CLR SUBFLG ; INIT SUBTRACT FLAG
    RTS PC ; RETURN

SUBFLG: .WORD
RELEB: SCOPE
        MOV PC, R2
        ADD #12, R2
        MOV @RELOC, PC ; GO RELOCATE PROGRAM CODE
RELEB: .WORD 0
; 8888888888888888 LAST ADDRESS OF CODE TO BE RELOCATED 888888888888
```

8013  
8014  
8015  
8016  
8017 035324  
8018 035324 000240  
8019 035326 112737 000105 001202  
8020 035334 113777 001202 143700  
8021 035342 005037 001550  
8022 035346 012704 000100  
8023 035352 032737 000400 001532  
8024 035360 001002  
8025 035362 000167 000206  
8026 035366 122777 000200 143656 1S:  
8027 035374 001374  
8028 035376 012737 001567 001276  
8029 035404 106277 143642  
8030 035410 000001  
8031  
8032  
8033 035412  
8034  
8035  
8036  
8037  
8038

\*\*\*\*\*  
;TEST 105 TELETYPE AND CLOCK TESTS  
\*\*\*\*\*  
†ST105:

NOP  
MOV8 #105,2#STSTNM  
MOV8 2#STSTNM,2SWR  
TTYCHK: CLR 2#FACTOR  
MOV #100,R4 ;SET R4 = CONSTANT 100  
BIT #TTOPT,2#OPT.CP ;BRANCH IF TTY  
BNE 1S ;ON SYSTEM  
JMP ARBFIN ;JUMP IF NOT  
1S: CMPB #200,2STPS ;CHECK IF TTY IS READY  
BNE 1S  
MOV #NULLS-1,2#SREG5 ;SET ADDRESS OF ASCII STRING TO TYPE  
ASRB 2STPS ;SET IE BIT. SEE TPISR FOR INT SERVICE.  
WAIT ;WAIT FOR INTERRUPT

DUMMY:  
;ROUTINE TO CHECK PRIORITY ARBITRATION LOGIC  
;THE BELOW TEST WILL INHIBIT INTERRUPTS ON LEVEL 6 AND BELOW (LOCKING  
;OUT THE LINE CLOCK) AND THEN SET UP THE TTY TO INTERRUPT. NEXT THE  
;PRIORITY LEVEL WILL BE SET TO 0 ALLOWING INTERRUPTS IN WHICH CASE  
;THE LINE CLOCK (AT LEVEL 6) SHOULD INTERRUPT BEFORE THE TTY (AT LEVEL 4).



```

8039 035412 132737 000020 177776 1$: BITB #20, @#PSW
8040 035420 001072 BNE ARBEX ;EXIT TEST IF 'T' BIT SET
8041 035422 030477 143624 2$: BIT R4, @STPS ;WAIT FOR TTY TO BE NOT
8042 035426 001375 BNE 2$ ;BUSY
8043 035430 112737 000300 177776 MOVB #300, @#PSW ;SET PRIORITY LEVEL 6
8044 035436 150477 143610 3$: BISB R4, @STPS ;SET IE BIT
8045 035442 100375 BPL 3$ ;AND WAIT FOR READY
8046 035444 032737 001000 001532 BIT #LKOPT, @#OPT.CP ;LINE CLOCK AVAILABLE?
8047 035452 001450 BEQ ARBFIN ;BRANCH IF NO
8048 035454 012737 035546 000064 MOV #7$, @#TPVEC ;SET TTY VECTOR
8049 035462 012737 035560 000100 MOV #8$, @#LKVEC ;SET CLOCK VECTORS
8050 035470 012737 000340 000102 MOV #PR7, @#LKVEC+2
8051 035476 005027 CLR (PC)+ ;CLEAR CHECK WORD
8052 035500 000000 4$: .WORD 0
8053 035502 000240 NOP
8054 035504 000240 NOP
8055 035506 000240 NOP
8056 035510 010437 177546 MOV R4, @#LKS
8057 035514 113700 5$: MOVB @#(PC)+, R0 ;GET CLOCK STATUS & BRANCH IF READY
8058 035516 177546 6$: .WORD LKS ;CONTAINS ADDRESS OF L CLOCK STAT
8059 035520 100375 BPL 5$
8060 035522 000240 NOP
8061 CLRB @#PSW ;AT THIS TIME BOTH THE CLOCK
8062 035524 105037 177776 ;ARE READY TO INTERRUPT
8063 ;A CLOCK INTERRUPT WILL OCCUR (8$) AND LOC 4$ WILL BE INCREMENTED
8064 ;AFTER THE CLOCK SERVICE A TTY INTERRUPT WILL OCCUR. THE TTY INT SERV
8065 ;ICE WILL SHIFT LEFT 4$.
8066
8067 035530 000240 NOP ;LEAVE TIME FOR INTERRUPTS
8068 035532 022767 000002 177740 CMP #2, 4$ ;CHECK THAT THE CLOCK
8069 035540 001415 BEQ ARBFIN ;& TTY INTERRUPTED IN
8070 035542 104000 HLT ;THE PROPER SEQUENCE
8071 035544 000413 BR ARBFIN
8072
8073 035546 005077 143500 7$: CLR @STPS ;CLEAR IE BIT
8074 035552 006367 177722 ASL 4$ ;SHIFT INDICATOR
8075 035556 000002 RTI ;RETURN
8076
8077 035560 005267 177714 8$: INC 4$
8078 035564 012737 046630 000100 MOV #LKSRV, @#LKVEC ;SET CLOCK VECTORS
8079 035572 000002 RTI
8080
8081
8082 035574 012737 054364 000064 ARBFIN: MOV #TPISR, @#TPVEC ;RESTORE TTY VECTOR
8083 035602 005077 143444 CLR @STPS ;CLEAR IE BIT
8084 035606
8085
8086
8087
8088
8089
8090 035606 000240 TST106: NOP
8091 035610 112737 000106 001202 MOVB #106, @#STSTNM
8092 035616 113777 001202 143416 MOVB @#STSTNM, @SWR
8093
8094 ;FIRST SETUP SOME MEM. MGMT. REGISTERS

```

```

8095 035624 012700 077406      MOV      #77406,R0      ;SET CONSTANT=R/W UP 4K WORDS
8096 035630 010037 172300      MOV      R0,#KIPDR0    ;SET KIPDR0,1,2,3,& 7 R/W UP 4K WORDS
8097 035634 010037 172302      MOV      R0,#KIPDR1
8098 035640 010037 172304      MOV      R0,#KIPDR2
8099 035644 010037 172306      MOV      R0,#KIPDR3
8100 035650 010037 172310      MOV      R0,#KIPDR4
8101 035654 010037 172312      MOV      R0,#KIPDR5
8102 035660 010037 172316      MOV      R0,#KIPDR7
8103 035664 005037 172340      CLR      #KIPAR0
8104 035670 012737 000200 172342  MOV      #200,#KIPAR1
8105 035676 012737 000400 172344  MOV      #400,#KIPAR2
8106 035704 012737 007600 172356  MOV      #7600,#KIPAR7
8107 035712 005737 177776      TST      #PSW          ;ARE WE IN USER MODE?
8108 035716 100464          BMI      11$          ;BRANCH IF YES
8109 035720 005737 001624      TST      #MXMML0     ;HAS MXMML0 BEEN FOUND YET?
8110 035724 001061          BNE      11$          ;BRANCH IF YES
8111 035726 012737 000001 177572  MOV      #1,#SRO
8112          ;GET ADDRESS OF LAST MEMORY LOCATION ON THE SYSTEM
8113 035734 012737 000600 172346  MOV      #600,#KIPAR3 ;INITIALLY MAP PAR3 TO 12K
8114 035742 012737 036006 000004  MOV      #106,#ERRVEC ;EXIT TO 106 ON FIRST TIMEOUT
8115 035750 005037 000006          CLR      #ERRVEC+2   ;BE SURE IN KERNEL MODE
8116 035754 012700 060000          MOV      #60000,R0   ;SETUP STARTING VIRTUAL ADDR.
8117 035760 012701 100000          MOV      #100000,R1 ;SETUP VIRT. ADDR. THAT SELECTS PAR4
8118 035764 005720          SS:  TST      (R0)+     ;TEST MEMORY LOCATION
8119 035766 020001          CMP      R0,R1       ;DOES VIRT. ADDR. STILL SELECT PAR3
8120 035770 103775          BLO      SS          ;BRANCH IF IT DOES
8121 035772 062737 000200 172346  ADD      #200,#KIPAR3 ;MAP PAR3 UP TO THE NEXT 4K
8122 036000 012700 060000          MOV      #60000,R0   ;RESET R0
8123 036004 000767          BR       SS          ;CONTINUE TESTING EACH VIRT. ADDR.
8124          ;UNTIL A TIMEOUT OCCURS
8125 036006 062706 000004          10$:  ADD      #4,SP       ;CLEANUP THE STACK
8126 036012 162700 000002          SUB      #2,R0        ;GET VIRT. ADDR. OF "TIMEOUT" LOC.
8127 036016 010003          MOV      R0,R3        ;MOVE VIRT. ADDR. INTO R3
8128 036020 072327 177772          ASH      #-6,R3       ;SHIFT VIRT. ADDR. RIGHT SIX BITS
8129 036024 042703 177600          BIC      #177600,R3   ;CLEAR OFF THE TOP 9 BITS
8130 036030 063703 172346          ADD      #KIPAR3,R3  ;ADD THE PAF TO THE BLOCK NO.
8131 036034 005002          CLR      R2          ;CLEAR R2
8132 036036 073227 000006          ASHC     #6,R2        ;FORM TOP 12 BITS OF PHYSICAL ADDR.
8133 036042 042702 177774          BIC      #177774,R2  ;CLEAR ALL BUT BITS 0 & 1 IN R2
8134 036046 042700 177700          BIC      #177700,R0  ;CLEAR BITS <15:06> IN VIRT. ADDR.
8135 036052 050003          BIS      R0,R3        ;SET BITS <05:00> IN PHYSICAL ADDR.
8136 036054 010237 001622          MOV      R2,#MXMMHI  ;SAVE BITS 16 & 17 OF TIMEOUT ADDR.
8137 036060 010337 001624          MOV      R3,#MXMML0  ;SAVE BITS 15-0 OF TIMEOUT ADDR.
8138 036064 005037 177572          CLR      #SRO        ;TURN MEM. MGMT. OFF AGAIN
8139
8140 036070 032737 001000 001532 11$:  BIT      #LKOPT,#OPT.CP ;BRANCH IF NOT AVAIL
8141 036076 001411          BEQ      UBESET
8142 036100 012737 046630 000100  MOV      #LKSRV,#LKVEC
8143 036106 012737 000340 000102  MOV      #PR7,#LKVEC+2
8144 036114 052737 000100 177546  BIS      #100,#LKS    ;SET IE BIT
8145
8146          ;*****
8147          ;TURN ON THE UNIBUS EXERCISER IF PRESENT
8148 036122 105737 001532          UBESET: TSTB #OPT.CP ;IS UBE OPTION AVAILABLE?
8149 036126 100023          BPL      MBTSET      ;BRANCH IF NO
8150 036130 032777 010000 143104  BIT      #SW12,#SWR   ;INHIBIT UBE?

```



```

8151 036136 001017      BNE      MBTSET      ;BRANCH IF YES
8152 036140 122737 000060 001574  CMPB    #60,2#SUBPASS ;FIRST SUB-PASS?
8153 036146 001056      BNE      STMM        ;BRANCH IF NO
8154 036150 105737 001200      TSTB    2#SPASS      ;FIRST PASS?
8155 036154 001053      BNE      STMM        ;BRANCH IF NO
8156 036156 105737 001545      TSTB    2#MMON      ;MEM MGMT ON?
8157 036162 001050      BNE      STMM        ;BRANCH IF YES
8158 036164 004737 053646      JSR     PC,2#UBEINIT ;INITIALIZE UBE
8159 036170 012772 064545 000000      MOV     #64545,2(R2) ;START UBE
8160
8161 ;*****
8162 ;TURN ON THE MASS BUS TESTER IF PRESENT
8163 036176 032737 002000 001532 MBTSET: BIT    #MBTOPT,2#OPT.CP ;IS MBT AVAILABLE?
8164 036204 001437      BEQ     STMM        ;BRANCH IF NO
8165 036206 032777 000010 143026      BIT     #SW3,2SWR    ;INHIBIT MBT?
8166 036214 001033      BNE      STMM        ;BRANCH IF YES
8167 036216 122737 000060 001574  CMPB    #60,2#SUBPASS ;FIRST SUB-PASS?
8168 036224 001027      BNE      STMM        ;BRANCH IF NO
8169 036226 105737 001200      TSTB    2#SPASS      ;FIRST PASS?
8170 036232 001024      BNE      STMM        ;BRANCH IF NO
8171 036234 105737 001545      TSTB    2#MMON      ;MEM MGMT ON?
8172 036240 001021      BNE      STMM        ;BRANCH IF YES
8173 036242 052777 000047 144064 MBT1:  BIS    #47,2MBTTBL+12 ;CLEAR THE MBT
8174 036250 012777 000007 144056      MOV     #7,2MBTTBL+12 ;SELECT UNIT 7
8175 036256 005077 144042      CLR     2MBTTBL+2    ;CLEAR THE WORD COUNT
8176 036262 012777 046304 144054      MOV     #MBTSRV,2MBTTBL+22 ;SETUP INTERRUPT VECTOR
8177 036270 012777 000240 144050      MOV     #PR5,2MBTTBL+24 ;SET VECTOR PSW
8178 036276 112777 000161 144016      MOVB   #161,2MBTTBL ;START MBT
8179
8180 ;*****
8181 ;SBTTL STMM ROUTINE
8182 ;ROUTINE TO SET UP MEMORY MANAGEMENT TO RELOCATE PROGRAM CODE ABOVE 16K
8183 ;CHECK IF PROGRAM IS TO BE RELOCATED.
8184 ;SWE=1=NO RELOCATION
8185 ;*****
8186 036304 032777 000100 142730 STMM:  BIT    #SW6,2SWR    ;RELOCATION DISABLED?
8187 036312 001402      BEQ     3$          ;BRANCH IF NO
8188 036314 000167 002266      JMP     ENDM
8189
8190 ;THE PROGRAM IS GOING TO RELOCATE.
8191 ;RELOCATION WILL BE PERFORMED IN KERNEL MODE WITH PSW SET AT PRIORITY
8192 ;LEVEL 4 (TO PREVENT TTY INTERRUPT-WHICH CHANGES DATA IN PROGRAM)
8193 ;THE 'T' BIT IS CLEARED (IF SET). AFTER THE DATA HAS BEEN WRITTEN IT IS
8194 ;VERIFIED BEFORE EXECUTION.
8195 036320 013727 177776      3$:   MOV     2#PSW,(PC)+ ;SAVE CURRENT PSW
8196 036324 000000      OLDPSW: .WORD 0
8197 036326 012737 000200 177776      MOV     #PR4,2#PSW
8198 036334 004767 015552      JSR     PC,CLRTBIT ;GO CLEAR 'T' BIT IF SET
8199
8200 ;NOW SETUP MEMORY MANAGEMENT REGISTERS.
8201
8202 036340 005037 172340      CLR     2#KIPAR0 ;NOTE: THESE 2 INSTRUCTIONS EFFECTIVELY
8203 036344 012737 000200 172342      MOV     #200,2#KIPAR1 ;RELOCATE PROGRAM EXECUTION
8204 036352 012737 000400 172344      MOV     #400,2#KIPAR2
8205 036360 013737 001562 172346      MOV     2#NEXPAR,2#KIPAR3 ;SET UP KIPAR3 & KIPAR4 & 5
8206 036366 013737 172346 172350      MOV     2#KIPAR3,2#KIPAR4

```

8207	036374	062737	000200	172350
8208	036402	013737	172346	172352
8209	036410	062737	000400	172352
8210	036416	012737	007600	172356
8211				
8212	036424	012700	077406	
8213	036430	010037	177600	
8214	036434	010037	177602	
8215	036440	010037	177604	
8216	036444	010037	177616	
8217	036450	016737	143106	177640
8218	036456	013737	177640	177642
8219	036464	062737	000200	177642
8220	036472	013737	177640	177644
8221	036500	062737	000400	177644
8222	036506	013737	172356	177656
8223				
8224	036514	012737	000001	177572
8225	036522	110637	001545	
8226	036526	013767	001202	021202
8227	036534	012737	040604	000004
8228	036542	005037	000006	
8229	036546	012702	060000	
8230	036552	005000		
8231				
8232	036554	012703	137776	
8233	036560	010013		
8234	036562	012737	055064	000004
8235	036570	000137	036770	
8236				
8237				
8238				
8239				
8240				
8241				
8242				
8243				
8244				
8245				
8246				
8247				
8248				
8249				
8250	036574	032777	000100	142440
8251	036602	001067		
8252	036604	105737	001545	
8253	036610	001064		
8254	036612	013700	001554	
8255	036616	010005		
8256				
8257	036620	010203		
8258	036622	010204		
8259	036624	160004		
8260	036626	010437	001552	
8261	036632	005737	001550	
8262	036636	001004		

```

ADD #200,2#KIPAR4
MOV 2#KIPAR3,2#KIPAR5
ADD #400,2#KIPAR5
MOV #7600,2#KIPAR7;AND OF COUSE THE I/O PAGE
;NOW SETUP USER MEM MGMT REGISTERS
1s: MOV #77406,R0 ;LOAD R0 WITH VALUE FOR PDRS
MOV R0,2#UIPDR0 ;SET UP USER MEM MGMT REGS
MOV R0,2#UIPDR1
MOV R0,2#UIPDR2
MOV R0,2#UIPDR7
MOV NEXPAR,2#UIPAR0
MOV 2#UIPAR0,2#UIPAR1
ADD #200,2#UIPAR1
MOV 2#UIPAR0,2#UIPAR2
ADD #400,2#UIPAR2
MOV 2#KIPAR7,2#UIPAR7

MOV #1,2#SRO ;ENABLE MEM MGMT
MOVB SP,2#MMON ;SET MEM MGMT ON IND = ON
MOV 2#$TSTNM,ENDTAG ;LOAD LAST WORD XFERED
RETRY: MOV #ENDMEM,2#ERRVEC;SET TIME OUT TRAP VECTOR
CLR 2#ERRVEC+2
MOV #60000,R2 ;SETUP GENERAL REGISTERS
CLR R0 ;DATA WILL BE RELOCATED FROM
;ADDRESS IN R0 TO ADDRESS IN R2
MOV #137776,R3 ;GET 12K WORDS TO RELOCATE
MOV R0,(R3) ;TRAP TO ENDMEM IF INSUFFICIENT MEMORY
MOV #ERPRT,2#ERRVEC ;RESTORE ERROR TRAP VECTOR
JMP 2#IOMON

;*****
;SBTTL RELOCATION ROUTINE
;* THIS ROUTINE IS USED TO RELOCATE THE 9 SUBTESTS UP TO 28K.
;* IF RELOCATION BY AN I/O DEVICE IS SELECTED, CONTROL IS PASSED
;* TO THE I/O MONITOR.
;* ENTER WITH:
;* FRSTAD=PHYSICAL ADDRESS OF FIRST CODE
;* FACTOR=NUMBER OF BYTES ABOVE BASE CODE
;* R2 =LAST PHYSICAL ADDRESS OF THE SECTION
;* EXIT TO I/O MONITOR WITH:
;* OLDBASE=FIRST PHYSICAL ADDRESS TO BE RELOCATED
;* NMBASL =FIRST PHYSICAL ADDRESS TO RELOCATE TO
;* IOWC =TWO'S COMPLIMENT WORD COUNT
;*****
RELOC: BIT #SW6,2#SWR ;IS RELOCATION DISABLED?
BNE EXITRE ;BRANCH IF YES
TSTB 2#MMON ;IS MEMORY MGMT ON?
BNE EXITRE ;BRANCH IF YES
MOV 2#FRSTAD,R0 ;GET FIRST ADDRESS TO BE RELOCATED
MOV R0,R5
;LAST ADDRESS IS IN R2
MOV R2,R3 ;SAVE LAST ADDRESS
MOV R2,R4
SUB R0,R4 ;R4 NOW HAS BYTE COUNT
MOV R4,2#$FACTOR ;SAVE BYTE COUNT
TST 2#FACTOR ;FIRST RELOC IS TO ENDTAG+2
BNE 1S ;BRANCH IF NOT EXECUTING BASE CODE

```



```

8263 036640 010237 036766      MOV      R2, @RETPC      ;SAVE RETURN PC TO NEXT SECTION
8264 036644 013702 001556      MOV      @#FRSTMEM, R2 ;GET FIRST ADDRESS TO RELOCATE TO
8265 036650 060204      1$: ADD      R2, R4        ;R4 NOW CONTAINS LAST MEM ADDRESS
8266 036652 020437 001560      CMP      R4, @#LSTMEM  ;ENOUGH MEMORY?
8267 036656 101042      BHI      NOMEM         ;BRANCH IF NO
8268 036660 160204      SUB      R2, R4        ;R4 NOW HAS BYTE COUNT
8269 036662 005037 001550      CLR      @#FACTOR
8270 036666 032777 000400 142346      BIT      #SW8, @SWR    ;INHIBIT RELOC BY I/O DEVICE?
8271 036674 001014      BNE      RELNIO        ;BRANCH IF YES
8272 036676 010037 001602      MOV      R0, @#OLDBASE ;SAVE START ADDRESS
8273 036702 010237 001604      MOV      R2, @#NWBASL  ;SAVE NEW BASE ADDRESS
8274 036706 005037 001606      CLR      @#NWBASH
8275 036712 006204      ASR      R4            ;MAKE IT A WORD COUNT
8276 036714 005404      NEG      R4            ;GET TWO'S COMPLIMENT
8277 036716 010437 001610      MOV      R4, @#IOWC   ;SAVE R4 AS WORDCOUNT
8278 036722 000167 000120      JMP      ENTER2       ;GO TO I/O MONITOR
8279
8280 036726 012022      ;RELOCATE BY CPU-MEMORY MANAGEMENT OFF
8281 036730 020003      RELNIO: MOV     (R0)+, (R2)+ ;RELOCATE CODE
8282 036732 001375      CMP      R0, R3       ;DONE YET?
8283 036734 004737 054026      BNE      RELNIO       ;BRANCH IF NO
8284 036740 102010      JSR      PC, @#CHKDAT ;GO CHECK DATA
8285 036742 010037 001304      BVC      EXITRE
8286 036746 010237 001536      MOV      R0, @#STMP0  ;SAVE R0 FOR TYPEOUT
8287 036752 004737 053724      MOV      R2, @#VADR   ;SAVE R2
8288 036756 104006      JSR      PC, @#CNVADR ;CONVERT R2 TO A PHYSICAL ADR
8289 036760 000401      ERROR   6
8290 036762 010207      EXITRE: MOV    R2, PC   ;GO EXECUTE RELOCATED CODE
8291 036764 011707      NOMEM:  MOV    (PC), PC ;GO TO NEXT SECTION
8292 036766 000000      RETPC:  .WORD  0       ;CONTAINS PC OF NEXT SECTION
8293
8294 ;*****
8295 ;SBTTL I/O RELOCATION MONITOR
8296 ;* THIS ROUTINE IS USED TO SCHEDULE I/O DEVICES FOR SUBTEST
8297 ;* RELOCATION AND PROGRAM RELOCATION. THE I/O DEVICE UNIT
8298 ;* NUMBER IS DETERMINED, THE BUS ADDRESS CALCULATED, THE WORD
8299 ;* COUNT CALCULATED AND PASSED TO THE DEVICE HANDLER.
8300 ;*****
8300 036770 012737 036776 001212 10MON: MOV     @#SLPERR ;SETUP ERROR LOOP
8301 036776 005037 001602      1$: CLR      @#OLDBASE
8302 037002 013705 172346      MOV      @#KIPAR3, R5 ;SETUP R4 AND R5
8303 037006 005004      CLR      R4           ;TO FORM 18 BIT ADDRESS
8304 037010 073427 000006      ASHC    #6, R4        ;FORM 18 BIT ADDRESS
8305 037014 010537 001604      MOV      R5, @#NWBASL ;SAVE LOWER 16 BITS
8306 037020 010437 001606      MOV      R4, @#NWBASH ;SAVE UPPER 2 BITS
8307 037024 032777 000400 142210      BIT      #SW8, @SWR   ;RELOCATE VIA I/O?
8308 037032 001402      BEQ     2$           ;BRANCH IF YES
8309 037034 000167 001374      JMP     RELOCP       ;GO RELOCATE VIA CP
8310 037040 012737 174000 001610 2$: MOV     #174000, @#IOWC ;SET WORD COUNT TO 2K
8311 037046 005037 001310      ENTER2: CLR    @#STMP2
8312 037052 012737 177776 001620      MOV     #-2, @#RNTBINX ;SETUP RUN TABLE INDEX
8313 037060 005037 001304      CLR     @#STMP0
8314 037064 005002      CLR     R2           ;CLEAR LEGAL DEV FLAG
8315 037066 032777 000040 142146 41$: BIT     #SW5, @SWR   ;INHIBIT ROUND ROBIN?
8316 037074 001416      BEQ     50$         ;BRANCH IF NO
8317 037076 005737 001304      TST     @#STMP0     ;FLAG SET?
8318 037102 001027      BNE     43$         ;BRANCH IF YES

```

8319	037104	117737	142132	001614		MOVB	2SWR,2#DEVINDX		;GET DEVICE FROM SWITCHES
8320	037112	042737	177770	001614		BIC	#177770,2#DEVINDX		;MASK LOWER 3 BITS
8321	037120	006337	001614			ASL	2#DEVINDX		;ADJUST FOR WORD INDEX
8322	037124	005237	001304			INC	2#STMP0		;SET FLAG
8323	037130	000414				BR	43\$		;CONTINUE
8324	037132	012705	000010		50\$:	MOV	#10,R5		;SET SOB COUNT
8325	037136	022737	000016	001614	40\$:	CMP	#16,2#DEVINDX		;LAST DEVICE YET?
8326	037144	001003				BNE	42\$		;BRANCH IF NO
8327	037146	012737	177776	001614	48\$:	MOV	#-2,2#DEVINDX		;INIT DEVICE INDEX
8328	037154	062737	000002	001614	42\$:	ADD	#2,2#DEVINDX		;INCREMENT INDEX
8329	037162	013703	001614		43\$:	MOV	2#DEVINDX,R3		;GET INDEX
8330	037166	012737	000401	001306		MOV	#401,2#STMP1		;INIT UNIT MASK
8331	037174	012704	000010			MOV	#10,R4	;SET SOB	COUNT
8332	037200	133763	001306	001646	44\$:	BITB	2#STMP1,SYSSIZE(R3)		;IS THIS UNIT EXISTENT?
8333	037206	001405				BEQ	52\$		;BRANCH IF NO
8334	037210	005202				INC	R2		;SET LEGAL DEVICE FLAG
8335	037212	133763	001387	001647		BITB	2#STMP1+1,SYSSIZE+1(R3)		;HAS IT BEEN USED?
8336	037220	001520				BEQ	11\$		;BRANCH IF NO
8337	037222	006337	001306		52\$:	ASL	2#STMP1		;SELECT NEXT UNIT
8338	037226	077414				SOB	R4,44\$		;CONTINUE
8339	037230	005737	001304			TST	2#STMP0		;INHIBIT ROUND ROBIN?
8340	037234	001013				BNE	45\$		;BRANCH IF YES
8341	037236	077541				SOB	R5,40\$		;CONTINUE
8342	037240	005702				TST	R2		;ANY DEVICES AT ALL?
8343	037242	001442				BEQ	46\$		;BRANCH IF NO
8344	037244	012704	000010			MOV	#10,R4		;SET SOB COUNT
8345	037250	012701	001647			MOV	#SYSSIZE+1,R1		;GET ADR OF SIZE TABLE
8346	037254	105021			47\$:	CLRB	(R1)+		;CLEAR ALL USED BITS
8347	037256	005201				INC	R1		;IN ALL DEVICES
8348	037260	077403				SOB	R4,47\$		;CONTINUE
8349	037262	000701				BR	41\$		
8350	037264	005702			45\$:	TST	R2		;WAS IT A LEGAL DEVICE?
8351	037266	001403				BEQ	49\$		;BRANCH IF NO
8352	037270	105063	001647			CLRB	SYSSIZE+1(R3)		;CLEAR ALL USED BITS THIS DEV
8353	037274	000732				BR	43\$		
8354	037276	010367	000016		49\$:	MOV	R3,60\$		
8355	037302	062767	055236	000010		ADD	#MSGINX,60\$		;GEN MESSAGE ADR
8356	037310	017767	000004	000002		MOV	260\$,60\$		
8357	037316	104401				TYPE			
8358	037320	000000			60\$:	.WORD			
8359	037322	104401	037330			TYPE	,65\$		;TYPE ASCIZ STRING
8360	037326	000407				BR	64\$		;GET OVER THE ASCIZ
8361					::65\$:	.ASCIZ	/UNAVAILABLE/<CRLF>		
8362	037346				64\$:				
8363	037346	000637				BR	ENTER2		
8364	037350	105737	001546		46\$:	TSTB	2#QV		;ACT11?
8365	037354	001016				BNE	51\$		;BRANCH IF YES
8366	037356	005227	177777			INC	#-1		
8367	037362	001013				BNE	51\$		
8368	037364	104401	037372			TYPE	,67\$		;TYPE ASCIZ STRING
8369	037370	000410				BR	66\$		;GET OVER THE ASCIZ
8370					::67\$:	.ASCIZ	?NO I/O DEVICES?<CRLF>		
8371	037412				66\$:				
8372	037412	105737	001545		51\$:	TSTB	2#MMON		;MGMT ON?
8373	037416	001012				BNE	61\$		;BRANCH IF YES
8374	037420	013700	001602			MOV	2#OLDBASE,RO		;RESTORE RO



8375	037424	013702	001604		MOV	2#NWBASL,R2	;RESTORE R2	
8376	037430	013703	001552		MOV	2#SFACOR,R3	;GET RELOCATION FACTOR	
8377	037434	060003			ADD	R0,R3	;FORM LAST ADDRESS	
8378	037436	010005			MOV	R0,R5	;SETUP R5	
8379	037440	000167	177262		JMP	RELNIO	;GO RELOCATE WITH CP	
8380	037444	012702	060000	61\$:	MOV	#60000,R2	;SETUP REGISTERS	
8381	037450	012703	137776		MOV	#137776,R3	;WITH FROM	
8382	037454	005000			CLR	R0	;AND TO ADDRESS	
8383	037456	000137	040434		JMP	2#RELOCP	;RELOCATE VIA CP	
8384	037462	105763	001752	11\$:	TSTB	RP3HSTAT(R3)	;IS HANDLER BUSY?	
8385	037466	100405			BMI	8\$	;BRANCH IF NO	
8386	037470	005737	001304		TST	2#STMP0	;ROUND ROBIN?	
8387	037474	001372			BNE	11\$	;BRANCH IF NO	
8388	037476	000167	177364		JMP	41\$		
8389	037502	005763	001752	8\$:	TST	RP3HSTAT(R3)	;DID HANDLER FAIL?	
8390	037506	100005			BPL	62\$	;BRANCH IF NO	
8391	037510	005737	001304		TST	2#STMP0	;ROUND ROBIN	
8392	037514	001402			BEQ	62\$	;BRANCH IF YES	
8393	037516	000137	040414		JMP	2#15\$		
8394	037522	153763	001307	001647	62\$:	BISB	2#STMP1+1,SYSSIZE+1(R3)	;SET UNIT USED BIT
8395	037530	005002			CLR	R2		
8396	037532	006037	001306	30\$:	ROR	2#STMP1	;ENCODE THE BIT POSITION	
8397	037536	005202			INC	R2	;INTO A UNIT NUMBER	
8398	037540	103374			BCC	30\$		
8399	037542	005302			DEC	R2		
8400	037544	010237	001616		MOV	R2,2#UNITNO	;SAVE UNIT NUMBER	
8401	037550	013763	001610	001772	10\$:	MOV	2#IOWC,RP3HWC(R3)	;GIVE WORD COUNT TO HANDLER
8402	037556	010304			MOV	R3,R4		
8403	037560	072427	000003		ASH	#3,R4	;ENCODE DEVICE FOR RUNTABLE	
8404	037564	053704	001616		BIS	2#UNITNO,R4	;ENCODE UNIT NUMBER	
8405	037570	006304			ASL	R4		
8406	037572	062737	000002	001620	ADD	#2,2#RNTBINX	;INCREMENT RUN TABLE INDEX	
8407	037600	013702	001620		MOV	2#RNTBINX,R2	;GET RUN TABLE INDEX	
8408	037604	110462	001667		MOVB	R4,RUNTABL+1(R2)	;ENTER DEV & UNIT IN TABLE	
8409	037610	013763	001616	002066	MOV	2#UNITNO,RP3UNIT(R3)	;GIVE HANDLER UNIT NUMBER	
8410	037616	012737	000240	000012	MOV	#PRS,2#RESVEC+2	;SETUP RESERVED VECTOR PSW	
8411	037624	016337	002102	000010	MOV	RP3HANA(R3),2#RESVEC	;SETUP RESERVED VECTOR	
8412	037632	006303			ASL	R3	;ADJUST INDEX	
8413	037634	013763	001602	002006	MOV	2#OLDBASE,RP3OLD(R3)	;GIVE HANDLER OLD BASE ADDRESS	
8414	037642	013763	001604	002036	MOV	2#NWBASL,RP3NWL(R3)	;GIVE HANDLER	
8415	037650	013763	001606	002040	MOV	2#NWBASH,RP3NWA(R3)	;NEW BASE ADDRESS	
8416	037656	005063	002010		CLR	RP3OLD+2(R3)	;ENSURE OLD BASE HIGH IS CLR	
8417	037662	000010			CALL	HANDLER		
8418	037664	105737	001545		TSTB	2#MMON	;IS MEMORY MANAGEMENT ON?	
8419	037670	001416			BEQ	13\$	;BRANCH IF NO	
8420	037672	022737	000012	001620	CMP	#12,2#RNTBINX	;TRANSFERED 12K YET?	
8421	037700	001412			BEQ	13\$	;BRANCH IF YES	
8422	037702	062737	010000	001602	ADD	#10000,2#OLDBASE	;ADD 2K	
8423	037710	062737	010000	001604	ADD	#10000,2#NWBASL	;TO BASE	
8424	037716	005537	001606		ADC	2#NWBASH	;ADDRESSES	
8425	037722	000137	037066		JMP	2#41\$		
8426	037726	113705	001645	13\$:	MOVB	2#LTICKS+1,R5	;GET SECOND COUNT	
8427	037732	062705	000002		ADD	#2,R5	;INCREMENT BY TWO	
8428	037736	162705	000074		SUB	#60.,R5	;ENSURE RESULT IS 59 OR LESS	
8429	037742	100002			BPL	31\$		
8430	037744	062705	000074		ADD	#60.,R5	;COUNT WAS LESS THAN 58-RESTORE	

```

8431 037750 012700 000010 31S: MOV #10,R0 ;SET SOB COUNT
8432 037754 005002 CLR R2
8433 037756 005003 CLR R3
8434 037760 005004 CLR R4
8435 037762 066203 001752 14S: ADD RP3HSTAT(R2),R3 ;ADD ALL THE HANDLER
8436 037766 005504 ADC R4 ;STATUS WORDS. WHEN ALL
8437 037770 062702 000002 ADD #2,R2 ;TRANSFERS ARE FINISHED
8438 037774 077006 SOB R0,14S ;RESULT WILL BE 2000
8439 037776 006103 ROL R3 ;(WITHOUT ROTATE)
8440 040000 005504 ADC R4
8441 040002 022703 004000 CMP #4000,R3 ;ALL DONE?
8442 040006 001406 BEQ 32S ;BRANCH IF YES
8443 040010 123709 001645 CMPB #LICKS+1,R5 ;TWO SECONDS ELAPSED YET?
8444 040014 001355 BNE 31S ;BRANCH IF NO
8445 040016 104015 ERROR 15 ;DEVICE HUNG
8446 040020 000177 141166 JMP #SLPERR ;RESTART RELOCATION
8447 040024 005704 32S: TST R4 ;ANY DEVICE ERRORS?
8448 040026 001172 BNE 15S ;BRANCH IF YES
8449 040030 105737 001545 TSTB #MMON ;MEM MGMT ON?
8450 040034 001012 BNE 25S ;BRANCH IF YES
8451 040036 013705 001602 MOV #OLDBASE,R5 ;SETUP R5 FOR DATA CHECK
8452 040042 010500 MOV R5,R0
8453 040044 063700 001552 ADD #SFACOR,R0 ;GET LAST ADDRESS
8454 040050 013702 001604 MOV #NWBASL,R2 ;OF GOOD DATA
8455 040054 063702 001552 ADD #SFACOR,R2 ;GET LAST ADDRESS
8456 040060 000406 BR 22S ;OF DATA TO BE CHECKED
8457 040062 012700 057776 25S: MOV #57776,R0 ;CONTINUE
8458 040066 012702 137776 MOV #137776,R2 ;GET LAST ADR OF GOOD DATA
8459 040072 012705 002500 MOV #2500,R5 ;GET LAST ADR OF DATA TO BE CHECKED
8460 040076 004737 054026 22S: JSR PC,#CHKDAT ;DON'T CHECK FIRST 2000 LOCATIONS
8461 040102 102173 BVC EXIT ;GO CHECK DATA
8462 040104 005001 CLR R1 ;EXIT IF NO ERRORS
8463 040106 010037 001304 MOV R0,#STMP0
8464 040112 010237 001536 MOV R2,#VADR
8465 040116 010203 MOV R2,R3 ;SAVE ERROR ADDRESS
8466 040120 005004 CLR R4
8467 040122 105737 001545 TSTB #MMON ;IS MEM MGMT ON?
8468 040126 001406 BEQ 16S ;BRANCH IF NO
8469 040130 162703 010000 17S: SUB #10000,R3 ;SUBTRACT 2K FROM ERROR ADDRESS
8470 040134 100403 BMT 16S ;BRANCH IF BLOCK IS FOUND
8471 040136 062704 000002 ADD #2,R4 ;COUNT ONE MORE BLOCK
8472 040142 000772 BR 17S ;CONTINUE
8473 040144 116404 001667 16S: MOVB RUNTBL+1(R4),R4 ;R4 NOW CONTAINS INDEX OF ERROR FOR RUN TIME TABLE
8474 040150 042704 177400 BIC #177400,R4 ;GET DEVICE THAT FAILED
8475 040154 006004 ROR R4 ;ENSURE HIGH BYTE CLEAR
8476 040156 005005 CLR R5 ;THROW AWAY LSB
8477 040160 073427 177775 ASHC #-3,R4 ;ENSURE R5 CLEAR
8478 040164 010500 MOV R5,R0 ;GET UNIT NUMBER IN R5
8479 040166 072027 177764 ASH #-14,R0
8480 040172 042700 177770 BIC #177770,R0
8481 040176 010037 001312 MOV R0,#STMP3
8482 040202 010403 MOV R4,R3 ;AND DEVICE INDEX IN R4 & R3
8483 040204 010337 001310 MOV R3,#STMP2

```



```

8487 040210 012737 000001 001306      MOV      #1,2#STMP1      ;ENCODE 3 BIT UNIT NO INTO
8488 040216 162705 020000      19$:    SUB      #20000,R5      ;ONE BIT IN THE LOW BYTE OF STMP1
8489 040222 103403      BCS     18$              ;BRANCH IF DONE
8490 040224 006137 001306      ROL     2#STMP1         ;SELECT NEXT UNIT
8491 040230 000772      BR     19$              ;CONTINUE
8492 040232 012737 040350 001212 18$:    MOV      #20$,2#SLPERR  ;SETUP LOOP RETURN
8493 040240 005701      TST     R1              ;DEVICE ERROR?
8494 040242 001010      BNE     100$           ;BRANCH IF YES
8495 040244 104010      ERROR   10              ;DATA CHECK ERROR
8496 040246 105737 001545      TSTB   2#MMON          ;MGMT ON?
8497 040252 001002      BNE     70$              ;BRANCH IF YES
8498 040254 000137 037046      71$:    JMP      2#ENTER2
8499 040260 000137 036770      70$:    JMP      2#IOMON
8500 040264 104007      100$:   ERROR   7
8501 040266 042763 100000 001752  BIC     #BIT15,RP3HSTAT(R3) ;CLEAR THE ERROR
8502 040274 022703 000002      CMP     #2,R3           ;RK?
8503 040300 002405      BLT     90$              ;BRANCH IF RH
8504 040302 003016      BGT     92$              ;BRANCH IF RPO3
8505 040304 112777 000001 141670  MOVB   #1,2#RKCS       ;CLEAR RK
8506 040312 000412      BR     92$
8507
8508 040314 022703 000012      90$:    CMP     #12,R3         ;RS?
8509 040320 001004      BNE     91$              ;BRANCH IF NO
8510 040322 052777 000040 141740  BIS    #BITS,2#RSCS2    ;CLEAR RS
8511 040330 000403      BR     92$
8512
8513 040332 052777 000040 141670  91$:    BIS    #BITS,2#RP4CS2  ;CLEAR RP
8514
8515 040340      92$:
8516 040340 105737 001545      TSTB   2#MMON          ;MGMT ON?
8517 040344 001345      BNE     70$              ;BRANCH IF YES
8518 040346 000742      BR     71$
8519 040350 052763 000400 001752  20$:    BIS    #BIT8,RP3HSTAT(R3) ;SET REPEAT FLAG IN HANDLER
8520 040356 016337 002102 000010  MOV    RP3HANA(R3),2#RESVEC ;SETUP RESERVED INSTRUCTION VECTOR
8521 040364 000010      CALLHANDLER
8522 040366 105763 001752      21$:    TSTB   RP3HSTAT(R3)    ;HANDLER FINISHED?
8523 040372 100375      BPL     21$              ;BRANCH IF NO
8524 040374 005763 001752      TST    RP3HSTAT(R3)    ;ANY ERROR?
8525 040400 100714      BMI     18$              ;BRANCH IF YES
8526 040402 032777 001000 140632  BIT    #BIT9,2#SWR     ;STILL LOOPING?
8527 040410 001357      BNE     20$              ;BRANCH IF YES
8528 040412 000725      BR     100$+2          ;CONTINUE TEST
8529
8530 040414 005004      15$:    CLR     R4              ;THE FOLLOWING CODE HANDLES DEVICE ERROR ON RELOCATION
8531 040416 010601      MOV    SP,R1            ;SET INDEX
8532 040420 005764 001702      24$:    TST    RUNTRAK(R4)    ;SEARCH FOR DEVICE ERROR
8533 040424 100647      BMI     16$              ;BRANCH IF ERROR
8534 040426 062704 000002      ADD    #2,R4            ;INCREMENT INDEX
8535 040432 000772      BR     24$              ;CONTINUE SEARCH
8536
8537 040434 012022      :RELOCATE BY CPU-MEMORY MANAGEMENT ON
RELOCP: MOV    (R0)+,(R2)+    ;RELOCATE CODE
8538 040436 020302      CMP    R3,R2            ;DONE YET?
8539 040440 001375      BNE    RELOCP           ;BRANCH IF NO
8540 040442 012705 002500      MOV    #2500,R5
8541 040446 004737 054026      JSR    PC,2#CHKDAT     ;CHECK DATA
8542 040452 102007      BVC    EXIT
    
```

```

8543 040454 010037 001304
8544 040460 010237 001536
8545 040464 104006
8546 040466 000167 176042
8547 040472 105737 001545
8548 040476 001002
8549 040500 000137 036762
8550 040504 062737 000077 001562
8551 040512 013737 172346 172340
8552 040520 013737 172350 172342
8553 040526 013737 172352 172344
8554
8555
8556
8557
8558 040534 013700 172340
8559 040540 072027 177774
8560 040544 110037 001203
8561 040550 005037 177776
8562 040554 012706 001200
8563 040560 016746 175540
8564 040564 012746 005530
8565 040570 105737 001544
8566 040574 001402
8567 040576 012716 036304
8568 040602 000002
8569
8570
8571 040604 022626
8572 040606 005037 177572
8573
8574
8575
8576
8577 040612 012737 000600 001562
8578 040620 005737 003330
8579 040624 001403
8580 040626 012737 001600 001562
8581 040634 105037 001545
8582
8583
8584
8585
8586
8587
8588 040640
8589 040640 012737 055064 000004
8590 040646 005037 177776
8591 040652 004767 013234
8592 040656 012706 001200
8593 040662 032737 000400 001532
8594 040670 001404
8595 040672 032777 000100 140352
8596 040700 001374
8597 040702 105237 001574
8598 040706 113702 001574

```

```

MOV R0,2#STMPD
MOV R2,2#VADR
ERROR 6
JMP RETRY
EXIT: TSTB 2#MMON ; MEM MGMT ON?
      BNE .+6 ; BRANCH IF YES
      JMP 2#EXITRE
      ADD 277,2#NEXPAR ; SET VALUE FOR NEXT RELOCATION
      MOV 2#IPAR3,2#KIPAR0
      MOV 2#KIPAR4,2#KIPAR1
      MOV 2#KIPAR5,2#KIPAR2
*****
:PROGRAM IS NOW EXECUTING IN KERNEL MODE RELOCATED TO ADDRESS AS SPEC-
:IFIED IN KIPAR0. FOR EX. IF KIPAR0=1600 THEN PROGRAM EXECUTING AT
:ADDRESS 16000+(PC)
      MOV 2#KIPAR0,R0 ; GET PAR0
      ASH 2-4,R0 ; GET BITS <11:4> IN LOW BYTE
      MOVB R0,2#STSTM+1 ; PUT IN DSPLY REG HIGH BYTE
      CLR 2#PSW
      MOV 2#KERSTK,SP ; SET KERNEL STACK PTR
      MOV OLDPSW,-(SP) ; RESTORE OLD PSW
      MOV 2#LOOP,-(SP)
      TSTB 2#NEXEC ; BRANCH IF TEST CODE TO
      BEQ 1$ ; BE EXECUTED
1$: MOV 2#STMM,(SP) ; RESTART PROGRAM AT LOOP
      RTI
:WHEN RELOCATION ABOVE 28K IS COMPLETE PROGRAM TRAPS TO ENDMEM.
ENDMEM: CMP (SP)+,(SP)+ ; POP STACK TWICE
ENDM: CLR 2#SRO ; DISABLE MEM MGMT
*****
:AT THIS TIME A 'SUB-PASS' HAS BEEN COMPLETED.
:PROGRAM NOW EXECUTING IN KERNEL MODE AT PC AS SHOWN (NO RELOCATION)
      MOV 2#600,2#NEXPAR ; RESET NEXT VALUE FOR PAR REGISTERS
      TST 2#PROT
      BEQ 2$
      MOV 2#1600,2#NEXPAR
2$: CLRB 2#MMON ; SET MEM MGMT ON IND = OFF
*****
.SBttl END OF SUB-PASS ROUTINE
:* THIS ROUTINE SETS UP THE PSW FROM THE PSW TABLE
:* FOR THE NEXT SUB-PASS. IT THEN STARTS THE PRINTER
:* (IF NOT ON ACT11) FOR TYPING THE END OF SUB-PASS MESSAGE.
*****
END:
END1: MOV 2#ERPRT,2#ERRVEC
      CLR 2#PSW ; CLEAR MODE BITS IN PSW
      JSR PC,CLRTBIT ; GO CLEAR 'T' BIT IF SET
      MOV 2#KERSTK,SP ; SET KERNEL STACK PTR
      BIT 2#BIT8,2#OPT.CP ; TTY AVAILABLE?
      BEQ 1$ ; BRANCH IF NO
      BIT 2#100,2#STPS ; CHECK IF OUTPUT DEVICE IS BUSY
      BNE .-6 ; IS AVAILABLE
1$: INCB 2#SUBPASS
      MOVB 2#SUBPASS,R2

```



```

8599 040712 162702 000060 SUB #60,R2
8600 040716 022702 000004 CMP #4,R2 ;END OF TEST?
8601 040722 001007 BNE 2$ ;BRANCH IF NOT AT END
8602 040724 012737 000060 001574 MOV #60,2$SUBPASS ;INIT SUBPASS COUNT TO ASCII 0
8603 040732 005046 CLR -(SP)
8604 040734 012746 041036 MOV #SEOP,-(SP)
8605 040740 000002 RTI
8606 040742 006302 2$: ASL R2
8607 040744 012737 001534 000014 MOV #SRTM,2$TBITVEC ;SET 'T' TRAP VECTOR
8608 040752 012737 001573 001276 MOV #SUBPASS-1,2$SREGS
8609 040760 106277 140266 ASRB 2$TPS
8610 040764 016246 055202 MOV PSWTAB(2),-(SP) ;PUSH NEXT PASS PSW ON STACK
8611 040770 012746 005530 MOV #LOOP,-(SP) ;RESTART PROGRAM AT LOOP
8612 040774 105737 001546 3$: TSTB 2$QV ;QV PASS?
8613 041000 001015 BNE RTI1 ;BRANCH IF YES
8614 041002 032737 000400 001532 BIT #BITB,2$OPT.CP ;TTY AVAILABLE?
8615 041010 001411 BEQ RTI1 ;BRANCH IF NO
8616 041012 122777 000200 140232 CMPB #200,2$TPS ;IS PRINTER READY?
8617 041020 001365 BNE 3$ ;BRANCH IF NO
8618 041022 012737 056217 001276 MOV #MSG20-1,2$SREGS
8619 041030 106277 140216 ASRB 2$TPS ;TYPE END SUBPASS MESSAGE
8620 041034 000002 RTI1: RTI ;RESTART PROGRAM AT LOOP WITH NEW PSW
; (FROM TABLE NEAR END OF PROGRAM)

```

.SBTTL END OF PASS ROUTINE

```

;*****
;INCREMENT THE PASS NUMBER ($PASS)
;TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
;WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
;IF THERES A MONITOR GO TO IT
;IF THERE ISN'T JUMP TO LOOP

```

```

8631 SEOP:
8632 041036 JSR PC,2$TYPTIME
8633 041036 004737 051010 CLR $TSTNM ;: ZERO THE TEST NUMBER
8634 041042 005067 140134 CLR $TIMES ;: ZERO THE NUMBER OF ITERATIONS
8635 041046 005067 140252 INC $PASS ;: INCREMENT THE PASS NUMBER
8636 041052 005267 140122 BIC #100000,$PASS ;: DON'T ALLOW A NEG. NUMBER
8637 041056 042767 100000 140114 DEC (PC)+ ;: LOOP?
8638 041064 005327 SEOPCT: .WORD 1
8639 041066 000001 BGT $DOAGN ;: YES
8640 041070 003063 MOV (PC)+,2(PC)+ ;: RESTORE COUNTER
8641 041072 012737 SENDCT: .WORD 1
8642 041074 000001 SEOPCT
8643 041076 041066 TYPE #65$ ;: TYPE ASCIZ STRING
8644 041100 104401 041106 BR #64$ ;: GET OVER THE ASCIZ
8645 041104 000407 ;:65$: .ASCIZ <12><15>/END PASS #/
8646 ;:64$:
8647 041124 MOV $PASS,-(SP) ;: SAVE $PASS FOR TYPEOUT
8648 041124 016746 140050 ;: TYPE PASS NUMBER
8649 TYPDS ;: GO TYPE--DECIMAL ASCII WITH SIGN
8650 041130 104405 TYPE #67$ ;: TYPE ASCIZ STRING
8651 041132 104401 041140 BR #66$ ;: GET OVER THE ASCIZ
8652 041136 000421 ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
8653 ;:66$:
8654 041202

```

```

8655 041202 016746 140006          MOV      SERTTL,-(SP)      ;; SAVE SERTTL FOR TYPEOUT
8656                                ;; TOTAL NUMBER OF ERRORS
8657 041206 104405          TYPDS   TYPE          SCRLF      ;; GO TYPE--DECIMAL ASCII WITH SIGN
8658 041210 104401 001335          TYPE          SERTTL      ;; TYPE CARRIAGE RETURN, LINE FEED
8659 041214 005067 137774          CLR      SERTTL      ;; CLEAR ERROR TOTAL
8660 041220 013700 000042          $GET42: MOV     @#42,RO  ;; GET MONITOR ADDRESS
8661 041224 001405          BEQ     SDOAGN        ;; BRANCH IF NO MONITOR
8662 041226 000005          RESET                                ;; CLEAR THE WORLD
8663 041230 004710          $ENDAD: JSR    PC,(RO)  ;; GO TO MONITOR
8664 041232 000240          NOP                                ;; SAVE ROOM
8665 041234 000240          NOP                                ;; FOR
8666 041236 000240          NOP                                ;; ACT11
8667 041240          SDOAGN:
8668 041240 000137          JMP     @PC)+         ;; RETURN
8669 041242 005530          $RTNAD: .WORD  LOOP
8670 041244 377 000          $ENULL: .BYTE  -1,-1,0 ;; NULL CHARACTER STRING
8671          .EVEN
8672          ;;*****
8673          ;;SRTTL RP11/RP03 HANDLER
8674          ;;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8675          ;;*****
8676 041250 104406          RP3DRV: SAVREG
8677 041252 105037 001752          CLRB    @RP3HSTA      ;; CLEAR DONE FLAG
8678 041256 032737 000400 001752          BIT     @BIT8,@RP3HSTA ;; REPEAT FLAG SET?
8679 041264 001403          BEQ     @S           ;; BRANCH IF NO
8680 041266 104407          RESREG
8681 041270 000137 043130          JMP     @RP3RPT
8682 041274 013737 001620 001630 8S:  MOV     @RNTBINX,@RP311 ;; SAVE RUN TABLE INDEX
8683 041302 032777 000020 137732          BIT     @SW4,@SWR    ;; INHIBIT RND DSK ADR?
8684 041310 001403          BEQ     @S           ;; BRANCH IF NO
8685 041312 005000          CLR    RO
8686 041314 005001          CLR    R1
8687 041316 000410          BR     @S
8688 041320 004737 052752          1S:  JSR    PC,@SRAND   ;; GO GET RANDOM NUMBER
8689 041324 013700 001566          MOV     @SHINUM,RO   ;; GET HI NUMBER
8690 041330 013701 001564          MOV     @SLONUM,R1   ;; GET LO NUMBER
8691 041334 073027 177771          ASHC   @-7,RO        ;; ADJUST TO FORM CYL ADR
8692 041340 042700 177000          4S:  BIC    @177000,RO   ;; GET RID OF UNUSED BITS
8693 041344 022700 000624          CMP     @624,RO      ;; LEGAL CYL?
8694 041350 100003          BPL    @S           ;; BRANCH IF YES
8695 041352 062700 000624          ADD     @624,RO      ;; MAKE IT LEGAL
8696 041356 000770          BR     @S
8697 041360 013702 001630          5S:  MOV     @RP311,R2    ;; GET RUN TABLE INDEX
8698 041364 016203 001666          MOV     RUNTBL(R2),R3 ;; GET DEVICE ID
8699 041370 042703 000777          BIC    @777,R3       ;; ID ONLY
8700 041374 050300          BIS    R3,RO         ;; COMBINE WITH CYL ADR
8701 041376 010062 001666          MOV     RO,RUNTBL(R2) ;; PUT BACK IN TABLE
8702 041402 072127 177775          ASH    @-3,R1        ;; GEN TRK-SECT ADR
8703 041406 010103          MOV     R1,R3        ;; SAVE
8704 041410 042701 160377          6S:  BIC    @160377,R1   ;; GET RID OF ALL BUT TRK
8705 041414 022701 011400          CMP     @11400,R1    ;; LEGAL TRAK?
8706 041420 100003          BPL    @S           ;; BRANCH IF YES
8707 041422 062701 011400          ADD     @11400,R1    ;; MAKE IT LEGAL
8708 041426 000770          BR     @S
8709 041430 042703 177760          2S:  BIC    @177760,R3   ;; GET SECTOR ADR
8710 041434 022703 000011          CMP     @11,R3      ;; IS IT LEGAL?

```



```

8711 041440 100003          BPL      3$          ;BRANCH IF YES
8712 041442 062703 000011  ADD      #11,R3     ;MAKE IT LEGAL
8713 041446 000770          BR       2$
8714 041450 050301          3$:  BIS   R3,R1     ;COMBINE TRK-SECT
8715 041452 010162 001702  MOV     R1,RUNTRAK(R2) ;PUT IN TABLE
8716 041456 010037 002120  MOV     RO,@#RP3HDC    ;SAVE DESIRED CYL
8717 041462 010137 002166  MOV     R1,@#RP3DA     ;SAVE DSK ADR
8718 041466 112737 177775 002146  MOV8    #-3,@#RP3TRY   ;INIT TRY COUNT
8719 041474 012737 000103 001626  7$:  MOV     #103,@#RP310 ;GET FUNCTION
8720 041502 013700 002010  MOV     @#RP3OLD+2,RO  ;GET BAE BITS
8721 041506 072027 000004  ASH    #4,RO         ;SHIFT TO BITS 4 & 5
8722 041512 050037 001626  BIS     RO,@#RP310    ;COMBINE WITH FUNCTION
8723 041516 010037 002010  MOV     RO,@#RP3OLD+2
8724 041522 013700 002066  MOV     @#RP3UNIT,RO
8725 041526 072027 000010  ASH    #10,RO        ;SHIFT UNIT NO TO RIGHT BITS
8726 041532 050037 001626  BIS     RO,@#RP310    ;COMBINE WITH FUNC & BAE
8727 041536 010037 002066  MOV     RO,@#RP3UNIT
8728 041542 104407          RESREG
8729 041544 053777 002066 140406  RP3WTRY: BIS @#RP3UNIT,@#RP3CS ;SET UNIT BITS
8730 041552 004737 041612          JSR     PC,@#LDRP3    ;LOAD RP3 REGISTERS
8731 041556 012777 043160 140406  MOV     @#RP3SRV,@#RP3VEC ;SET VECTOR
8732 041564 005077 140404          CLR    @#RP3PSW
8733 041570 005037 002134          CLR    @#RP3FUN
8734 041574 005777 140354          1$:  TST   @#RP3DS
8735 041600 100375          BPL    1$
8736 041602 013777 001626 140350  MOV     @#RP310,@#RP3CS
8737 041610 000002          RTI
8738 041612 013777 002116 140346  LDRP3: MOV @#RP3HDA,@#RP3DA ;LOAD DSK ADR
8739 041620 013777 002120 140342  MOV     @#RP3HDC,@#RP3DC ;LOAD CYL ADR
8740 041626 013777 001772 140326  MOV     @#RP3HWC,@#RP3WC ;LOAD WORD COUNT
8741 041634 013777 002006 140322  MOV     @#RP3OLD,@#RP3BA ;LOAD BUS ADR
8742 041642 000207          RTS     PC           ;RETURN
8743
8744
8745 ;:*****
8746 ;.SBTTL RK11/RK05 HANDLER
8747 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8748 ;:*****
8748 041644 104406          RKDRV: SAVREG
8749 041646 105037 001754          CLRB   @#RKHSTAT     ;CLEAR DONE FLAG IN HANDLER STAT
8750 041652 032737 000400 001754  BIT     #BIT8,@#RKHSTAT ;REPEAT FLAG SET?
8751 041660 001403          BEQ    5$           ;BRANCH IF NO
8752 041662 104407          RESREG
8753 041664 000137 043700          JMP    @#RKRP3
8754 041670 013737 001620 001634  5$:  MOV     @#RNTBINX,@#RK11 ;SAVE RUN TABLE INDEX
8755 041676 105037 001754          CLRB   @#RKHSTAT     ;CLEAR DONE FLAG IN HANDLER STAT
8756 041702 032777 000020 137332  BIT     #SW4,@SWR     ;RANDOM DSK ADDRESS?
8757 041710 001403          BEQ    6$           ;BRANCH IF YES
8758 041712 005000          CLR    RO           ;CLEAR REGISTERS
8759 041714 005001          CLR    R1
8760 041716 000406          BR     7$
8761 041720 004737 052752          6$:  JSR     PC,@#$RAND    ;FOR ADDRESS CHECKING
8762 041724 013700 001566          MOV     @#$HINUM,RO  ;GET RANDOM NUMBER
8763 041730 013701 001564          MOV     @#$LONUM,R1  ;GET HIGH NUMBER
8764 041734 072027 177775          7$:  ASH    #-3,RO       ;GET LO NUMBER
8765          ;ADJUST TO FORM
8766 041740 010001          MOV     RO,R1        ;CYLINDER ADDRESS
                        ;SAVE IN R1

```

```

8767 041742 042701 160037      4S:   BIC      #160037,R1      ;GET RID OF SURF-SECT BITS
8768 041746 022701 014300      CMP      #14300,R1    ;IS IT A LEGAL TRAK?
8769 041752 100003          BPL      3S          ;BRANCH IF YES
8770 041754 062701 014340      ADD      #14340,R1    ;ADD MAXIMUM TRAK
8771 041760 000770          BR       4S          ;TRY AGAIN
8772
8773 041762 072127 177773      3S:   ASH      #-5,R1      ;ADJUST CYLINDER ADDRESS
8774 041766 013702 001634      MOV      @#RK11,R2    ;GET RUN TABLE INDEX
8775 041772 016203 001666      DWORRY: MOV     RUNTBL(R2),R3
8776 041776 042703 000777      BIC      #777,R3
8777 042002 050103          BIS      R1,R3
8778 042004 010362 001666      MOV      R3,RUNTBL(R2) ;ENTER CYLINDER ADR IN RUN TABLE
8779 042010 072027 177770      ASH      #-10,R0     ;GENER SECTOR-SURF ADDRESS
8780 042014 042700 177740      BIC      #177740,R0   ;GET RID OF EXTRA BITS
8781 042020 010003          MOV      R0,R3       ;SAVE
8782 042022 042700 000020      BIC      #814,R0     ;GET RID OF SURFACE BIT
8783 042026 022700 000012      CMP      #12,R0     ;IS SECTOR ADDRESS LEGAL?
8784 042032 100004          BPL      1S          ;BRANCH IF YES
8785 042034 062700 000012      ADD      #12,R0     ;MAKE IT LEGAL
8786 042040 042700 000020      BIC      #814,R0     ;GET RID OF CARRY FROM ADD
8787 042044 042703 000017      1S:   BIC      #17,R3     ;GET SURFACE ADDRESS
8788 042050 050300          BIS      R3,R0     ;GENER COMP SECT-SURF ADDRESS
8789 042052 010062 001702      MOV      R0,RUNTRAK(R2) ;SAVE IN RUN TRAK TABLE
8790 042056 072127 000005      ASH      #5,R1      ;ADJUST CYLINDER ADDRESS
8791 042062 050100          BIS      R1,R0     ;CONCATINATE TRK & SECT ADDR
8792 042064 013701 002070      MOV      @#RKUNIT,R1 ;GET UNIT NUMBER
8793 042070 072127 000015      ASH      #15,R1     ;ADJUST
8794 042074 050100          BIS      R1,R0     ;CONCATINATE UNIT, TRK, SURF, SECT
8795 042076 010037 002122      MOV      R0,@#RKHDA  ;SAVE
8796 042102 112737 177775 002147      MOVVB   #-3,@#RKTRY  ;SET RETRY COUNT
8797 042110 012767 000103 137514 2S:   MOV      #103,RK10   ;SET FUNCTION
8798 042116 013700 002014      MOV      @#RKOLD+2,R0 ;GET BA EXTENDED
8799 042122 072027 000004      ASH      #4,R0     ;ADJUST
8800 042126 050037 001632      BIS      R0,@#RK10  ;PUT IN WITH FUNCTION
8801 042132 010037 002014      MOV      R0,@#RKOLD+2 ;SAVE IN MEMORY
8802 042136 104407
8803 042140 013777 002122 140042  RKWTRY: MOV     @#RKHDA,@#RKDA ;LOAD DISK ADDRESS
8804 042146 013777 001774 140030      MOV      @#RKHWC,@#RKWC ;LOAD WORD COUNT
8805 042154 013777 002012 140024      MOV      @#RKOLD,@#RKBA ;LOAD BUS ADDRESS
8806 042162 012777 043730 140022      MOV      #RKSrv,@#RKVEC ;LOAD INTERRUPT VECTOR
8807 042170 005077 140020      CLR      @#RKPSW
8808 042174 005037 002136      CLR      @#RKFUN
8809 042200 032777 000100 137770      BIT      #BIT6,@#RKDS ;SET FUNCTION TO WRITE
8810 042206 001774          BEQ      -6         ;UNIT READY?
8811 042210 013777 001632 137764      MOV      @#RK10,@#RKCS ;LOAD FUNCTION AND GO
8812 042216 000006          RTT
8813
8814
8815 ;*****
8816 ;SBTTL RH11/RP04 HANDLER
8817 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
8818 ;*****
8818 042220 104406      RP4DRV: SAVREG
8819 042222 105037 001762      CLR      @#RP4HSTA    ;CLEAR DONE FLAG
8820 042226 032737 000400 001762      BIT      #BIT8,@#RP4HST ;REPEAT FLAG SET?
8821 042234 001403          BEQ      6S         ;BRANCH IF NO
8822 042236 104407          RESREG

```



8823	042240	000137	044504			JMP	2#RP4RPT		
8824	042244	013737	001620	001636	6\$:	MOV	2#RNTBINX, 2#RP411		:SAVE RUN TABLE INDEX
8825	042252	105037	001762			CLRB	2#RP4HSTA		:CLEAR DONE FLAG
8826	042256	032777	000020	136756		BIT	#SW4, 2#SWR		:RANDOM DSK ADDRESS?
8827	042264	001403				BEQ	1\$		:BRANCH IF YES
8828	042266	005000				CLR	R0		
8829	042270	005001				CLR	R1		
8830	042272	000410				BR	4\$		
8831	042274	004737	052752		1\$:	JSR	PC, 2#SRAND		:GET RANDOM NUMBER
8832	042300	013700	001566			MOV	2#SHINUM, R0		:GET HI NUMBER
8833	042304	013701	001564			MOV	2#SLONUM, R1		:GET LO NUMBER
8834	042310	073027	177771			ASHC	#-7, R0		:ADJUST TO FORM CYL. ADR.
8835	042314	042700	177000		4\$:	BIC	#177000, R0		:GET RID OF UNUSED BITS
8836	042320	022700	000631			CMP	#631, R0		:LEGAL CYLINDER
8837	042324	100003				BPL	5\$		:BRANCH IF YES
8838	042326	062700	000631			ADD	#631, R0		:MAKE IT LEGAL
8839	042332	000770				BR	4\$		
8840									
8841	042334	013702	001636		5\$:	MOV	2#RP411, R2		:GET RUN TABLE INDEX
8842	042340	016203	001666			MOV	RUNTB(R2), R3		:GET DEVICE ID
8843	042344	042703	000777			BIC	#777, R3		:SAVE ID ONLY
8844	042350	050003				BIS	R0, R3		:COMBINE WITH CYL ADR
8845	042352	010362	001666			MOV	R3, RUNTB(R2)		:PUT IN RUN TABLE
8846	042356	072127	177775			ASH	#-3, R1		:GEN TRAK-SECT ADR
8847	042362	042701	160340			BIC	#160340, R1		:GET RID OF UNUSED BITS
8848	042366	010103				MOV	R1, R3		:SAVE
8849	042370	042701	000037			BIC	#37, R1		:GET RID OF SECT BITS
8850	042374	022701	011000			CMP	#11000, R1		:LEGAL TRAK?
8851	042400	100004				BPL	2\$		:BRANCH IF YES
8852	042402	062701	011000			ADD	#11000, R1		:MAKE IT LEGAL
8853	042406	042701	020000			BIC	#BIT13, R1		:GET RID OF ADD CARRY
8854	042412	042703	177740		2\$:	BIC	#177740, R3		:GET SECTOR ADR
8855	042416	022703	000025			CMP	#25, R3		:LEGAL SECTOR
8856	042422	100004				BPL	3\$		:BRANCH IF YES
8857	042424	062703	000025			ADD	#25, R3		:MAKE IT LEGAL
8858	042430	042703	000040			BIC	#BITS, R3		:GET RID OF ADD CARRY
8859	042434	050301			3\$:	BIS	R3, R1		:COMBINE TRACK-SECTOR
8860	042436	010162	001702			MOV	R1, RUNTRAK(R2)		:PUT TRAK-SECT IN TABLE
8861	042442	010037	002130			MOV	R0, 2#RP4HDC		:SAVE CYLINDER ADR
8862	042446	010137	002126			MOV	R1, 2#RP4HDA		:SAVE TRAK-SECTOR ADR
8863	042452	112737	177775	002151		MOVB	#-3, 2#RP4TRY		:SET TRY COUNT
8864	042460	104407				RESREG			
8865	042462								
8866	042462	004767	000026			JSR	PC, LDRP4		:LOAD RP4 REGISTERS
8867	042466	012777	044530	137556		MOV	#RP4SRV, 2#RP4VEC		:LOAD INTERRUPT VECTOR
8868	042474	005077	137554			CLR	2#RP4PSW		
8869	042500	005037	002142			CLR	2#RP4FUN		:SET FUNCTION TO WRITE
8870	042504	112777	000161	137504		MOVB	#161, 2#RP4CS1		:LOAD FUNCTION AND GO
8871	042512	000002				RTI			:RETURN
8872									
8873	042514	013777	002076	137506	LDRP4:	MOV	2#RP4UNIT, 2#RP4CS2		:LOAD UNIT NUMBER
8874	042522	012777	010000	137520		MOV	#BIT12, 2#RP4OF		:SET FORMAT TO 16 BIT
8875	042530	013777	002130	137502		MOV	2#RP4HDC, 2#RP4DC		:LOAD CYLINDER ADR
8876	042536	013777	002126	137462		MOV	2#RP4HDA, 2#RP4DA		:LOAD TRAK-SECTOR
8877	042544	013777	002002	137446		MOV	2#RP4HWC, 2#RP4WC		:LOAD WORD COUNT
8878	042552	010546				MOV	R5, -(SP)		:SAVE R5

```

8879 042554 013705 002030      MOV      @#RP4OLD+2,R5
8880 042560 072527 000010      ASH      #10,R5          ;SHIFT EXTENDED ADDR. BITS
8881 042564 042705 176377      BIC      #176377,R5      ;
8882 042570 042777 001400 137420  BIC      #1400,@#RP4CS1  ;AND
8883 042576 050577 137414      BIS      R5,@#RP4CS1    ;LOAD INTO RP4CS1
8884 042602 012605          MOV      (SP)+,R5        ;RESTORE R5
8885 042604 013777 002026 137410  MOV      @#RP4OLD,@#RP4BA ;LOAD BUS ADR
8886 042612 000207          RTS      PC              ;RETURN

```

```

8887
8888
8889
8890
8891
8892
8893
8894
8895
8896
8897
8898
8899
8900
8901
8902
8903
8904
8905
8906
8907
8908
8909
8910
8911
8912
8913
8914
8915
8916
8917
8918
8919
8920
8921
8922
8923
8924
8925
8926
8927
8928
8929
8930
8931
8932
8933
8934

```

```

;*****
;SBTTL RH11/RP04 HANDLER
;*      SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF HANDLER
;*****
RSDRV: SAVREG

```

```

      CLR      @#RSHSTAT      ;CLEAR DONE FLAG
      BIT      #BIT8,@#RSHSTAT ;REPEAT FLAG SET?
      BEQ      3$             ;BRANCH IF NO
      RESREG
      JMP      @#RSRPT
      MOV      @#RNTDINX,@#RS11 ;SAVE RUN TABLE INDEX
      BIT      #SW4,@#SWR      ;RANDOM DSK ADR?
      BEQ      1$             ;BRANCH IF YES
      CLR      R0
      CLR      R1
      BR      4$
      JSR      PC,@#$RAND      ;GET RANDOM NUMBER
      MOV      @#$HINUM,R0
      ASH      #-4,R0
      MOV      R0,R1          ;SAVE RANDOM NUMBER
      BIC      #170077,R0     ;GET TRACK ADR
      CMP      #7600,R0       ;IS IT LEGAL?
      BPL      5$             ;BRANCH IF YES
      ADD      #7600,R0       ;MAKE IT LEGAL
      BR      4$
      MOV      @#RS11,R2      ;GET RUN TABLE INDEX
      ASH      #-6,R0         ;ADJUST TRACK ADR
      MOV      R0,RUNTRK(R2)  ;SAVE TRAK ADR IN RUN TBL
      BIC      #177700,R1     ;GET SECTOR ADR
      CMP      #77,R1        ;IS IT LEGAL?
      BPL      2$             ;BRANCH IF YES
      ADD      #77,R1        ;MAKE IT LEGAL
      BR      6$
      MOV      R1,RUNTRK(R2)  ;SAVE IN RUN TRAK TABLE
      ASH      #6,R0
      BIS      R1,R0          ;COMBINE SECTOR TRAK
      MOV      R0,@#RSHDA     ;SAVE AS DSK ADR
      MOV      #-3,@#RSTRY    ;SET TRY COUNT
      RESREG
      JSR      PC,@#LDRS      ;GO LOAD REGISTERS
      MOV      @#RSSRV,@#RSVEC ;SET INTERRUPT VECTOR
      CLR      @#RSPSW
      CLR      @#RSFUN
      TSTB    @#RSDS          ;SET FUNCTION TO WRITE
      BEQ      1$             ;IS DRIVE READY?
      MOV      #161,@#RSCS1   ;BRANCH IF NO
      RTI

```



```

8935
8936 043044 013777 002100 137216 LDRS: MOV @#RSUNIT,@#RSCS2 ;LOAD UNIT NUMBER
8937 043052 013777 002132 137206 MOV @#RSHDA,@#RSDA ;LOAD DSK ADR
8938 043060 013777 002004 137172 MOV @#RSHWC,@#RSWC ;LOAD WORD COUNT
8939 043066 010546 MOV R5,-(SP) ;SAVE R5
8940 043070 013705 002034 MOV @#RSOLD+2,R5 ;SHIFT EXTENDED ADDR. BITS
8941 043074 072527 000010 ASH #10,R5 ; AND
8942 043100 042705 176377 BIC #176377,R5 ; LOAD
8943 043104 042777 001400 137144 BIC #1400,@#RSCS1
8944 043112 050577 137140 BIS R5,@#RSCS1 ;INTO RSCS1
8945 043116 012605 MOV (SP)+,R5 ;RESTORE R5
8946 043120 013777 002032 137134 MOV @#RSOLD,@#RSBA ;LOAD BUS ADDRESS
8947 043126 000207 RTS PC ;RETURN
8948
8949
8950 ;*****
8951 ;SBTTL RP11/RP03 SERVICE ROUTINE
8952 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
8953 ;*****
8954 043130 000005 RP3RPT: RESET
8955 043132 005337 002134 DEC @#RP3FUN ;RESTORE FUNCTION
8956 043136 022737 000001 002134 CMP #1,@#RP3FUN ;WHAT IS IT?
8957 043144 001472 BEQ RP31 ;BRANCH IF WC
8958 043146 002402 BLT 1$ ;BRANCH IF WRITE
8959 043150 000137 041544 JMP @#RP3WTRY ;BRANCH TO READ
8960 043154 000167 000366 1$: JMP RP33
8961 043160 005237 002134 RP3SRV: INC @#RP3FUN ;INCREMENT FUNCTION
8962 043164 022737 000002 002134 CMP #2,@#RP3FUN ;WHAT IS IT?
8963 043172 001501 BEQ RP3WCK ;BRANCH TO WRITE CHECK
8964 043174 100002 BPL +6
8965 043176 000137 043606 JMP @#RP3READ
8966
8967 ;FUNCTION JUST EXECUTED WAS A WRITE
8968 043202 032737 000400 001752 BIT #BIT0,@#RP3HSTAT ;REPEAT FLAG SET?
8969 043210 001036 BNE RP3LOOP ;BRANCH IF YES
8970 043212 005777 136742 TST @#RP3CS ;ANY ERRORS?
8971 043216 100045 BPL RP31 ;BRANCH IF NO
8972 043220 105737 002146 TSTB @#RP3TRY ;TRIED 3 TIMES?
8973 043224 001415 BEQ RP3ERR ;BRANCH IF YES
8974 043226 112777 000001 136724 MOVB #BIT0,@#RP3CS ;CLEAR THE DRIVE
8975 043234 105777 136720 TSTB @#RP3CS ;CONTROLLER READY?
8976 043240 100375 BPL -4 ;BRANCH IF NO
8977 043242 105237 002146 INCB @#RP3TRY ;INCREMENT TRY COUNT
8978 043246 013746 177776 MOV @#PSW,-(SP) ;MAINTAIN SAME PSW
8979 043252 012746 041544 MOV @#RP3WTRY,-(SP) ;SET RETRY ADDRESS
8980 043256 000002 RTI ;RETURN
8981 043260 012737 100200 001752 RP3ERR: MOV #100200,@#RP3HSTA ;SET ERROR BIT IN HAND. STA
8982 043266 010046 MOV RO,-(SP) ;SAVE RO
8983 043270 013700 001630 MOV @#RP311,RO ;GET RUNTABLE INDEX
8984 043274 052760 100000 001702 BIS #BIT15,RUNTRAK(RO) ;SET ERROR BIT
8985 043302 012600 MOV (SP)+,RO ;RESTORE RO
8986 043304 000002 RTI ;RETURN
8987 043306 012737 100200 001752 RP3LOOP: MOV #100200,@#RP3HSTAT ;SET DONE AND ERROR
8988 043314 005777 136640 TST @#RP3CS ;ANY ERRORS?
8989 043320 100403 BMI 1$ ;BRANCH IF YES
8990 043322 042737 100000 001752 BIC #BIT15,@#RP3HSTAT ;CLEAR ERROR BIT
    
```

```

8991 043330 000002          1S:      RTI                      ;RETURN
8992                                     :WRITE WAS OK- NOW DO A WRITE CHECK
8993 043332 112737 177775 002146 RP31:  MOVB  #3,2#RP3TRY          ;INIT TRY COUNT
8994 043340 012737 000107 001626      MOV  #107,2#RP310        ;SET FUNCTION
8995 043346 053737 002010 001626      BIS  2#RP3OLD+2,2#RP310 ;SET BAE BITS
8996 043354 053737 002066 001626      BIS  2#RP3UNIT,2#RP310 ;SET UNIT BITS
8997 043362 004737 041612          RP32:  JSR  PC,2#LDRP3      ;LOAD RP3 REGISTERS
8998 043366 013777 001626 136564      MOV  2#RP310,2#RP3CS    ;LOAD FUNCTION AND GO
8999 043374 000002          RTI                      ;RETURN
9000
9001                                     :FUNCTION JUST EXECUTED WAS A WRITE CHECK
9002 043376 032737 000400 001752 RP3WCK: BIT  #BIT8,2#RP3HSTAT      ;REPEAT FLAG SET?
9003 043404 001340          BNE  RP3LOOP          ;BRANCH IF YES
9004 043406 005777 136546          TST  2#RP3CS          ;ANY ERRORS?
9005 043412 100031          BPL  1S              ;BRANCH IF NO
9006 043414 005737 001630          TST  2#RP311         ;FIRST 2K?
9007 043420 001422          BEQ  4S              ;BRANCH IF YES
9008 043422 105737 002146          5S:  TSTB 2#RP3TRY      ;TRIED 3 TIMES?
9009 043426 001714          BEQ  RP3ERR         ;BRANCH IF YES
9010 043430 005337 002134          DEC  2#RP3FUN        ;RESTORE FUNCTION
9011 043434 112777 000001 136516      MOVB  #BIT0,2#RP3CS   ;CLEAR THE DRIVE
9012 043442 105777 136512          TSTB 2#RP3CS        ;CONTROLLER READY?
9013 043446 100375          BPL  -4              ;BRANCH IF NO
9014 043450 105237 002146          INCB 2#RP3TRY        ;INCREMENT TRY COUNT
9015 043454 013746 177776          MOV  2#PSW,-(SP)
9016 043460 012746 043362          MOV  #RP32,-(SP)
9017 043464 000002          RTI
9018 043466 032777 000010 136462 4S:  BIT  #BIT3,2#RP3ER    ;GO TRY AGAIN
9019 043474 001752          BEQ  5S              ;WRITE CHECK ERROR?
9020                                     ;BRANCH IF NO
9021                                     :WRITE CHECK OK- NOW DO A READ
9022 043476 112737 177775 002146 1S:  MOVB  #3,2#RP3TRY          ;RESTORE TRY COUNT
9023 043504 010046          2S:  MOV  RO,-(SP)      ;SAVE RO
9024 043506 013700 002040          MOV  2#RP3NWH,RO     ;GET BAE BITS
9025 043512 072027 000004          ASH  #4,RO           ;ADJUST
9026 043516 010037 002040          MOV  RO,2#RP3NWH     ;SAVE
9027 043522 012600          MOV  (SP)+,RO        ;RESTORE RO
9028 043524 012737 000105 001626      MOV  #105,2#RP310    ;SET FUNCTION
9029 043532 053737 002040 001626      BIS  2#RP3NWH,2#RP310 ;SET BAE BITS
9030 043540 053737 002066 001626      BIS  2#RP3UNIT,2#RP310 ;SET UNIT NUMBER
9031 043546 013777 002116 136412 RP33: MOV  2#RP3HDA,2#RP3DA ;LOAD DSK ADR
9032 043554 013777 002120 136406      MOV  2#RP3HDC,2#RP3DC ;LOAD CYL
9033 043562 013777 001772 136372      MOV  2#RP3HWC,2#RP3WC ;LOAD WORD COUNT
9034 043570 013777 002036 136366      MOV  2#RP3NWL,2#RP3BA ;LOAD BUS ADR
9035 043576 013777 001626 136354      MOV  2#RP310,2#RP3CS ;LOAD FUNCTION AND GO
9036 043604 000002          RTI                      ;RETURN
9037
9038                                     :FUNCTION JUST EXECUTED WAS A READ
9039 043606 032737 000400 001752 RP3READ: BIT #BIT8,2#RP3HSTAT ;REPEAT FLAG SET?
9040 043614 001234          BNE  RP3LOOP        ;BRANCH IF YES
9041 043616 005777 136336          TST  2#RP3CS        ;ANY ERRORS?
9042 043622 100022          BPL  1S              ;BRANCH IF NO
9043 043624 105737 002146          TSTB 2#RP3TRY      ;TRIED 3 TIMES?
9044 043630 001613          BEQ  RP3ERR         ;BRANCH IF YES
9045 043632 005337 002134          DEC  2#RP3FUN        ;RESTORE FUNCTION
9046 043636 112777 000001 136314      MOVB  #BIT0,2#RP3CS   ;CLEAR THE DRIVE

```



```

9047 043644 105777 136310          TSTB  @RP3CS          ;CONTROLLER READY?
9048 043650 100375                BPL    -4             ;BRANCH OF NO
9049 043652 105237 002146          INCB  @#RP3TRY       ;INCREMENT TRY COUNT
9050 043656 013746 177776          MOV   @#PSW,-(SP)
9051 043662 012746 043546          MOV   @#RP33,-(SP)
9052 043666 000002                RTI
9053 043670 112737 000200 001752 1$:  MOVB  #200,@#RP3HSTA ;GO TRY AGAIN
9054 043676 000002                RTI                ;SET DONE FLAG
9055                                     ;RETURN
9056                                     ;*****
9057 .SBTTL RK11/RK05 SERVICE ROUTINE
9058 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
9059 ;*****
9059 043700 000005          RKRPT: RESET
9060 043702 005337 002136          DEC   @#RKFUN        ;RESTORE FUNCTION
9061 043706 022737 000001 002136  CMP   #1,@#RKFUN     ;WHAT IS IT?
9062 043714 001475                BEQ   RK1            ;BRANCH IF WC
9063 043716 002402                BLT   1$            ;BRANCH IF WRITE
9064 043720 000137 042140          JMP   @#RKWTRY       ;IT WAS A WRITE
9065 043724 000137 044332 1$:      JMP   @#RK3
9066 043730 062737 000001 002136  RKSrv: ADD  #1,@#RKFUN ;FIND OUT WHAT FUNCTION
9067                                     ;WAS EXECUTED
9068 043736 022737 000002 002136  CMP   #2,@#RKFUN     ;WAS IT A WRITE CHECK?
9069 043744 001507                BEQ   RKWRCK        ;BRANCH IF YES
9070 043746 100002                BPL   +6             ;BRANCH IF IT WAS A WRITE
9071 043750 000137 044364          JMP   @#RKREAD
9072
9073                                     ;FUNCTION JUST EXECUTED WAS A WRITE. ANY ERRORS?
9074 043754 032737 000400 001754  BIT   #BIT8,@#RKHSTAT ;REPEAT FLAG SET?
9075 043762 001040                BNE   RKLOOP        ;BRANCH IF YES
9076 043764 005777 136212          TST   @#RKCS        ;ANY ERRORS?
9077 043770 100047                BPL   RK1            ;BRANCH IF NO
9078 043772 105737 002147          TSTB  @#RKTRY       ;TRIED 3 TIMES?
9079 043776 001417                BEQ   RKERR         ;BRANCH IF YES
9080 044000 012777 000001 136174  MOV   #1,@#RKCS     ;CLEAR THE ERROR
9081 044006 004737 044466          JSR   PC,@#TIMER    ;WAIT A LITTLE
9082 044012 105777 136164          TSTB  @#RKCS        ;WAIT FOR CONT CLR TO FINISH
9083 044016 100375                BPL   -4             ;INCREMENT TRY COUNT
9084 044020 105237 002147          INCB  @#RKTRY
9085 044024 013746 177776          MOV   @#PSW,-(SP)
9086 044030 012746 042140          MOV   @#RKWTRY,-(SP)
9087 044034 000002                RTI
9088 044036 012737 100200 001754  RKERR: MOV  #100200,@#RKHSTAT ;SET ERROR & DONE FLAG
9089 044044 010046                MOV  RO,-(SP)        ;SAVE RO
9090 044046 013700 001634          MOV  @#RK11,RO      ;GET SAVED RUN TABLE INDEX
9091 044052 052760 100000 001702  BIS  #BIT15,RUNTRAK(RO) ;SET ERROR BIT IN RUN TABLE
9092 044060 012600                MOV  (SP)+,RO       ;RESTORE RO
9093 044062 000002                RTI                ;RETURN
9094
9095 044064 012737 100200 001754  RKLOOP:MOV #100200,@#RKHSTAT ;SET DONE AND ERROR BITS
9096 044072 005777 136104          TST   @#RKCS        ;ANY ERRORS?
9097 044076 100403                BMI   1$            ;BRANCH IF YES
9098 044100 042737 100000 001754  BIC  #BIT15,@#RKHSTAT ;CLEAR ERROR BIT
9099 044106 000002                RTI                ;RETURN
9100                                     ;WRITE WAS OK, NOW DO A WRITE CHECK
9101 044110 112737 177775 002147  RK1:  MOVB  #-3,@#RKTRY ;RESTORE TRY COUNT
9102 044116 012767 000507 135506  MOV   #507,RK10     ;SET FUNCTION TO WRITE

```

9103	044124	053767	002014	135500		BIS	@RKOLD+2,RK10		;SET BA EXT BITS
9104	044132	013777	002122	136050	RK2:	MOV	@RKHDA,@RKDA		;LOAD DISK ADDRESS
9105	044140	013777	001774	136036		MOV	@RKHWC,@RKWC		;LOAD WORD COUNT
9106	044146	013777	002012	136032		MOV	@RKOLD,@RKBA		;LOAD BUS ADDRESS
9107	044154	016777	135452	136020		MOV	RK10,@RKCS		;START FUNCTION
9108	044162	000002				RTI			;RETURN
9109									
9110									;FUNCTION JUST EXECUTED WAS A WRITE CHECK. ANY ERRORS?



J15

9111	044164	032737	000400	001754	RKWRCK:BIT	#BIT8, @RKHSTAT	:REPEAT FLAG SET?
9112	044172	001334			BNE	RKLOOP	:BRANCH IF YES
9113	044174	005777	136002		TST	@RKCS	:ANY ERRORS?
9114	044200	100033			BPL	1\$	:BRANCH IF NO
9115	044202	005737	001634		TST	@R'11	:FIRST 2K?
9116	044206	001424			BEQ	4\$	:BRANCH IF YES
9117	044210	105737	002147	5\$:	TSTB	@RKTRY	:TRIED 3 TIMES?
9118	044214	001710			BEQ	RKERR	:BRANCH IF YES
9119	044216	005337	002136		DEC	@RKFUN	:SET FUNCTION BACK TO WC
9120	044222	012777	000001	135752	MOV	#1, @RKCS	:CLEAR THE ERROR
9121	044230	004737	044466		JSR	PC, @TIMER	:WAIT A LITTLE

9122	044234	105777	135742		TSTB	ARKCS		;WAIT FOR CLR TO FINISH
9123	044240	100375			BPL	-4		
9124	044242	105237	002147		INCB	ARKTRY		;INCREMENT TRY COUNT
9125	044246	013746	177776		MOV	PSW, -(SP)		
9126	044252	012746	044132		MOV	ARK2, -(SP)		
9127	044256	000002			RTI			
9128	044260	032777	040000	135714	4S:	BIT	BIT14, ARKCS	;HARD ERROR?
9129	044266	001350			BNE	5S		;BRANCH IF YES
9130								
9131								
9132	044270	112737	177775	002147	1S:	WRITE CHECK WAS OK, NOW DO A READ.		
9133	044276	010046			2S:	MOVB	-3, ARKTRY	;RESTORE TRY COUNT
9134	044300	013700	002044			MOV	RO, -(SP)	;SAVE RO
9135	044304	072027	000004			MOV	ARKNEWH, RO	;GET BA EXT
						ASH	4, RO	;ADJUST



```

9136 044310 010037 002044      MOV      RO, @RKNEWH      ;SAVE
9137 044314 012600      MOV      (SP)+, RO      ;RESTORE RO
9138 044316 012767 000105 135306  MOV      #105, RK10     ;SET FUNCTION
9139 044324 053767 002044 135300  BIS      @RKNEWH, RK10  ;SET BA EXT BITS IN FUNCTION
9140 044332 013777 002122 135650  RK3:    MOV      @RKHDA, @RKDA  ;LOAD DISK ADDRESS
9141 044340 013777 001774 135636  MOV      @RKHWC, @RKWC  ;LOAD WORD COUNT
9142 044346 013777 002042 135632  MOV      @RKNEWL, @RKBA ;LOAD BUS ADDRESS
9143 044354 016777 135252 135620  MOV      RK10, @RKCS   ;LOAD FUNCTION AND GO
9144 044362 000002      RTI                      ;RETURN
9145
9146      ;FUNCTION JUST EXECUTED WAS A READ. ANY ERRORS?
9147 044364 032737 000400 001754  RKREAD: BIT      @BIT8, @RKHSTAT ;REPEAT FLAG SET?
9148 044372 001234      BNE      RKLOOP        ;BRANCH IF YES
9149 044374 005777 135602      TST      @RKCS        ;ANY ERRORS?
9150 044400 100026      BPL      1$           ;BRANCH IF NO
9151 044402 105737 002147      TSTB    @RKTRY        ;TRIED 3 TIMES?
9152 044406 001002      BNE      3$           ;BRANCH IF NO
9153 044410 000167 177422      JMP      RKERR
9154 044414 005337 002136 3$:      DEC      @RKFUN      ;SET FUNCTION BACK TO READ
9155 044420 012777 000001 135554  MOV      #1, @RKCS    ;CLEAR THE ERROR
9156 044426 004737 044466      JSR     PC, @TIMER    ;WAIT A LITTLE
9157 044432 105777 135544      TSTB    @RKCS        ;WAIT FOR CLR TO FINISH
9158 044436 100375      BPL     -4
9159 044440 105237 002147      INCB    @RKTRY      ;INCREMENT TRY COUNT
9160 044444 013746 177776      MOV     @PSW, -(SP)
9161 044450 012746 044332      MOV     @RK3, -(SP)
9162 044454 000002      RTI
9163 044456 112737 000200 001754  1$:    MOV     @200, @RKHSTAT ;SET DON E FLAG
9164 044464 000002      RTI                      ;RETURN
9165 044466 005067 000010  TIMER: CLR      1$
9166 044472 105267 000004  2$:    INCB    1$
9167 044476 001375      BNE     2$
9168 044500 000207      RTS     PC
9169 044502 000000  1$:    .WORD
9170
9171      ;*****
9172      ;SBTTL  RH11/RP04 SERVICE ROUTINE
9173      ;*    SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
9174      ;*****
9175 044504 000005  RP4RPT: RESET
9176 044506 005337 002142      DEC     @RP4FUN      ;RESTORE FUNCTION
9177 044512 022737 000001 002142  CMP     #1, @RP4FUN  ;WHAT IS IT?
9178 044520 001501      BEQ     RP41         ;BRANCH IF WC
9179 044522 002560      BLT     RP43         ;BRANCH IF READ
9180 044524 000137 042462      JMP     @RP4WTRY    ;GO TO WRITE
9181 044530 005237 002142  RP4SRV: INC     @RP4FUN ;FIND OUT WHAT FUNCTION
9182 044534 022737 000002 002142  CMP     #2, @RP4FUN ;WAS JUST EXECUTED
9183 044542 001504      BEQ     RP4WCK
9184 044544 100576      BMI     RP4READ
9185
9186      ;WRITE FUNCTION WAS JUST EXECUTED.
9187 044546 032737 000400 001762  BIT     @BIT8, @RP4HSTAT ;REPEAT FLAG SET?
9188 044554 001050      BNE     RP4LOOP    ;BRANCH IF YES
9189 044556 032777 040000 135450  BIT     @BIT14, @RP4DS ;ANY ERRORS
9190 044564 001457      BEQ     RP41         ;BRANCH IF NO
9191 044566 105737 002151      TSTB   @RP4TRY    ;TRIED 3 TIMES?

```

M15

```

9192 044572 001426          BEQ      RP4ERR          ;BRANCH IF YES
9193 044574 052777 000040 135426    BIS      #BIT5,2#RP4CS2 ;CLEAR ALL ERRORS
9194 044602 004737 042514          JSR      PC,2#LDRP4     ;RELOAD THE UNIT NO
9195 044606 105237 002151          INCB    2#RP4TRY        ;INCREMENT TRY COUNT
9196 044612 013746 177776          MOV     2#PSW,-(SP)    ;SETUP THE STACK TO
9197 044616 012746 042462          MOV     2#RP4HTRY,-(SP);TRY WRITE AGAIN
9198 044622 032737 000400 001762    BIT      #BIT8,2#RP4HSTAT;REPEAT FLAG SET?
9199 044630 001006          BNE     2$             ;BRANCH IF YES
9200 044632 012777 000007 135356    MOV     #7,2#RP4CS1    ;RECALIBRATE
9201 044640 105777 135352    1$:     TSTB    2#RP4CS1  ;DRIVE READY?
9202 044644 100375          BPL     1$             ;BRANCH IF NO
9203 044646 000002          2$:     RTI              ;
9204 044650 012737 100200 001762    RP4ERR: MOV     #100200,2#RP4HSTA ;SET ERROR & DONE BIT
9205 044656 010046          MOV     RO,-(SP)       ;SAVE RO
9206 044660 013700 001636          MOV     2#RP4I1,RO    ;GET RUN TABLE INDEX
9207 044664 052760 100000 001702    BIS      #BIT15,RUNTRAK(RO);SET ERROR BIT
9208 044672 012600          MOV     (SP)+,RO      ;RESTORE RO
9209 044674 000002          RTI                    ;RETURN
9210
9211 044676 012737 100200 001762    RP4LOOP:MOV #100200,2#RP4HSTAT ;SET DONE AND ERROR BITS
9212 044704 032777 040000 135322    BIT      #BIT14,2#RP4DS ;ANY ERRORS?
9213 044712 001003          BNE     1$             ;BRANCH IF YES
9214 044714 042737 100000 001762    BIC      #BIT15,2#RP4HSTAT ;CLEAR ERROR BIT
9215 044722 000002          1$:     RTI              ;RETURN
9216
9217 044724 112737 177775 002151    :WRITE OK...NOW DO A WRITE CHECK.
9218 044732 105777 135276    RP41:   MOV     #-3,2#RP4TRY ;INITIALIZE TRY COUNT
9219 044736 001775          RP42:   TSTB    2#RP4DS    ;IS DRIVE READY?
9220 044740 004737 042514          BEQ     RP42           ;BRANCH IF NO
9221 044744 112777 000151 135244    JSR     PC,2#LDRP4     ;LOAD FUNCTION AND GO
9222 044752 000002          MOV     #151,2#RP4CS1
9223
9224          :FUNCTION JUST EXECUTED WAS A WRITE CHECK
9225 044754 032737 000400 001762    RP4WCK: BIT      #BIT8,2#RP4HSTAT ;REPEAT FLAG SET?
9226 044762 001345          BNE     RP4LOOP       ;BRANCH IF YES
9227 044764 032777 040000 135242    BIT      #BIT14,2#RP4DS ;ANY ERRORS?
9228 044772 001421          BEQ     1$             ;BRANCH IF NO
9229 044774 105737 002151          3$:     TSTB    2#RP4TRY  ;TRIED 3 TIMES?
9230 045000 001723          BEQ     RP4ERR        ;BRANCH IF YES
9231 045002 005337 002136          DEC     2#RKFUN        ;SET FUNCTION TO WC
9232 045006 052777 000040 135214    BIS      #BIT5,2#RP4CS2 ;CLEAR ALL ERRORS
9233 045014 004737 042514          JSR     PC,2#LDRP4     ;RELOAD THE UNIT NO
9234 045020 105237 002151          INCB    2#RP4TRY        ;INCREMENT TRY COUNT
9235 045024 013746 177776          MOV     2#PSW,-(SP)
9236 045030 012746 044732          MOV     2#RP4,-(SP)
9237 045034 000002          RTI                    ;TRY AGAIN
9238 045036 032777 040000 135164    1$:     BIT      #BIT14,2#RP4CS2 ;WRITE CHECK ERROR?
9239 045044 001404          BEQ     2$             ;BRANCH IF NO
9240 045046 005737 001636          TST     2#RP4I1        ;FIRST 2K?
9241 045052 001401          BEQ     2$             ;
9242 045054 000747          BR      3$             ;
9243
9244          :WRITE CHECK WAS OK...NOW DO A READ.
9245 045056 112737 177775 002151    2$:     MOV     #-3,2#RP4TRY ;INITIALIZE TRY COUNT
9246 045064 105777 135144    RP43:   TSTB    2#RP4DS    ;IS DRIVE READY?
9247 045070 001775          BEQ     RP43          ;BRANCH IF NO
    
```



```

9248 045072 004737 042514 JSR PC, @LDRP4 ;LOAD REGISTERS
9249 045076 010546 MOV RS, -(SP) ;SAVE RS
9250 045100 013705 002060 MOV @RP4NH, RS ;SHIFT EXTENDED
9251 045104 072527 000010 ASH #10, RS ;ADDRESS BITS
9252 045110 042777 001400 135100 BIC #1400, @RP4CS1 ;AND
9253 045116 050577 135074 BIS RS, @RP4CS1 ;LOAD INTO RP4CS1
9254 045122 012605 MOV (SP)+, RS ;RESTORE RS
9255 045124 013777 002056 135070 MOV @RP4NL, @RP4BA ;LOAD BUS ADR
9256 045132 112777 000171 135056 MOVB #171, @RP4CS1 ;LOAD FUNCTION AND GO
9257 045140 000002 RTI ;RETURN
9258
9259 ;FUNCTION JUST EXECUTED WAS A READ.
9260 045142 032737 000400 001762 RP4READ: BIT #BIT8, @RP4HSTAT ;REPEAT FLAG SET?
9261 045150 001252 BNE RP4LOOP ;BRANCH IF YES
9262 045152 032777 040000 135054 BIT #BIT14, @RP4DS ;ANY ERRORS?
9263 045160 001421 BEQ IS ;BRANCH IF NO
9264 045162 105737 002151 TSTB @RP4TRY ;TRIED 3 TIMES?
9265 045166 001630 BEQ RP4ERR ;BRANCH IF YES
9266 045170 005337 002142 DEC @RP4FUN ;SET FUNCTION TO A READ
9267 045174 052777 000040 135026 BIS #BITS, @RP4CS2 ;CLEAR ALL ERRORS
9268 045202 004737 042514 JSR PC, @LDRP4 ;RELOAD THE UNIT NO
9269 045206 105237 002151 INCB @RP4TRY ;INCREMENT TRY COUNT
9270 045212 013746 177776 MOV @PSW, -(SP)
9271 045216 012746 045064 MOV @RP43, -(SP)
9272 045222 000002 RTI ;TRY AGAIN
9273 045224 112737 000200 001762 IS: MOVB #200, @RP4HSTA ;SET DONE FLAG
9274 045232 000002 RTI ;RETURN
9275
9276 ;*****
9277 ;SBTTL RH11/RP04 SERVICE ROUTINE
9278 ;* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
9279 ;*****
9279 045234 000005 RSRPT: RESET
9280 045236 005337 002144 DEC @RSFUN ;RESTORE FUNCTION
9281 045242 022737 000001 002144 CMP #1, @RSFUN ;WHAT IS IT?
9282 045250 001465 BEQ RS41 ;BRANCH IF WC
9283 045252 002542 BLT RS43 ;BRANCH IF WRITE
9284 045254 000137 043004 JMP @RSWTRY
9285 045260 005237 002144 RSSRV: INC @RSFUN ;FIND OUT WHAT FUNCTION
9286 045264 022737 000002 002144 CMP #2, @RSFUN ;WAS JUST EXECUTED
9287 045272 001470 BEQ RSACK
9288 045274 100562 BMI RSREAD
9289
9290 ;WRITE FUNCTION WAS JUST EXECUTED
9291 045276 032737 000400 001764 BIT #BIT8, @RSHSTAT ;REPEAT FLAG SET?
9292 045304 001034 BNE RSLOOP ;BRANCH IF YES
9293 045306 032777 040000 134760 BIT #BIT14, @RSDS ;ANY ERRORS?
9294 045314 001443 BEQ RS41 ;BRANCH IF NO
9295 045316 105737 002152 TSTB @RSTRY ;TRIED 3 TIMES?
9296 045322 001412 BEQ RSERR ;BRANCH IF YES
9297 045324 052777 000040 134736 BIS #BITS, @RSCS2 ;CLEAR ALL ERRORS
9298 045332 105237 002152 INCB @RSTRY ;INCREMENT TRY COUNT
9299 045336 013746 177776 MOV @PSW, -(SP) ;SETUP THE STACK TO
9300 045342 012746 043004 MOV @RSWTRY, -(SP) ;TRY THE WRITE AGAIN
9301 045346 000002 RTI
9302 045350 012737 100200 001764 RSERR: MOV #100200, @RSHSTAT ;SET ERROR AND DONE BIT
9303 045356 010046 MOV RO, -(SP) ;SAVE RO

```

```

9304 045360 013700 001640      MOV      @RS11,RO      ;GET RUN TBL INDEX
9305 045364 052760 100000 001702  BIS      @BIT15,RUNTRAK(RO) ;SET ERROR BIT
9306 045372 012600      MOV      (SP)+,RO      ;RESTORE RO
9307 045374 000002      RTI
9308
9309 045376 012737 100200 001764 RSL00P: MOV      @100200,@RSHSTAT ;SET DONE AND ERROR BITS
9310 045404 032777 040000 134662  BIT      @BIT14,@RSDS ;ANY ERRORS?
9311 045412 001003      BNE     1$ ;BRANCH IF YES
9312 045414 042737 100000 001764  BIC      @BIT15,@RSHSTAT ;CLEAR ERROR BIT
9313 045422 000002      RTI ;RETURN
9314
9315 045424 112737 177775 002152  RS41:  MOVB   @-3,@RSTRY ;INIT TRY COUNT
9316 045432 105777 134636  RS42:  TSTB   @RSDS ;IS DRIVE READY?
9317 045436 001775      BEQ     RS42 ;BRANCH IF NO
9318 045440 004737 043044      JSR     PC,@LDRS ;LOAD RS REGISTERS
9319 045444 112777 000151 134604  MOVB   @151,@RSCS1 ;LOAD FUNCTION AND GO
9320 045452 000002      RTI ;RETURN
9321
9322      ;FUNCTION JUST EXECUTED WAS A WRITE CHECK
9323 045454 032737 000400 001764  RSWCK: BIT      @BIT8,@RSHSTAT ;REPEAT FLAG SET?
9324 045462 001345      BNE     RSL00P ;BRANCH IF YES
9325 045464 032777 040000 134602  BIT      @BIT14,@RSDS ;ANY ERRORS?
9326 045472 001417      BEQ     1$ ;BRANCH IF NO
9327 045474 105737 002152  3$:    TSTB   @RSTRY ;TRIED 3 TIMES?
9328 045500 001723      BEQ     RSERR ;BRANCH IF YES
9329 045502 005337 002144      DEC     @RSFUN ;SET FUNCTION BACK TO WC
9330 045506 052777 000040 134554  BIS      @BIT5,@RSCS2 ;CLEAR THE ERROR
9331 045514 105237 002152      INCB   @RSTRY ;INCREMENT THE TRY COUNT
9332 045520 013746 177776      MOV     @PSW,-(SP)
9333 045524 016746 177702      MOV     RS42,-(SP)
9334 045530 000002      RTI ;TRY AGAIN
9335
9336 045532 032777 040000 134530  1$:    BIT      @BIT14,@RSCS2 ;WRITE CHECK ERROR?
9337 045540 001404      BEQ     2$ ;BRANCH IF NO
9338 045542 005737 001640      TST    @RS11 ;FIRST 2K?
9339 045546 001401      BEQ     2$ ;BRANCH IF YES
9340 045550 000751      BR     3$
9341
9342      ;WRITE CHECK WAS OK...NOW DO A READ.
9343 045552 112737 177775 002152  2$:    MOVB   @-3,@RSTRY ;INIT TRY COUNT
9344 045560 105777 134510  RS43:  TSTB   @RSDS ;IS DRIVE READY?
9345 045564 001775      BEQ     RS4? ;BRANCH IF NO
9346 045566 004737 043044      JSR     PC,@LDRS ;LOAD RS REGISTERS
9347 045572 010546      MOV     RS,-(SP) ;SAVE RS
9348 045574 013705 002064      MOV     @RSNEWH,R5 ;SHIFT EXTENDED
9349 045600 072527 000010      ASH    @10,R5 ;ADDRESS BITS
9350 045604 042705 176377      BIC    @176377,R5 ;AND
9351 045610 042777 001400 134440  BIC    @1400,@RSCS1
9352 045616 050577 134434      BIS    R5,@RSCS1 ;LOAD INTO RSCS1
9353 045622 012605      MOV     (SP)+,R5 ;RESTORE R5
9354 045624 013777 002062 134430  MOV     @RSNEWL,@RSBA ;LOAD BUS ADR
9355 045632 112777 000171 134416  MOVB   @171,@RSCS1 ;LOAD FUNCTION AND GO
9356 045640 000002      RTI ;RETURN
9357
9358      ;FUNCTION JUST EXECUTED WAS A READ.
9359 045642 032737 000400 001764  RSREAD: BIT      @BIT8,@RSHSTAT ;REPEAT FLAG SET?
    
```



```

9360 045650 001252          BNE      RSLOOP          ;BRANCH IF YES
9361 045652 032777 040000 134414  BIT      #BIT14,RSDS    ;ANY ERRORS?
9362 045660 001417          BEQ      1$              ;BRANCH IF NO
9363 045662 105737 002152    TSTB    @RSTRY          ;TRIED 3 TIMES?
9364 045666 001630          BEQ      RSERR          ;BRANCH IF YES
9365 045670 005337 002144    DEC     @RSFUN          ;RESTORE FUN TO READ
9366 045674 052777 000040 134366  BIS     @BITS,RSRCS2   ;CLEAR ALL ERRORS
9367 045702 105237 002152    INCB   @RSTRY          ;INCREMENT TRY COUNT
9368 045706 013746 177776    MOV     @PSW,-(SP)
9369 045712 012746 045560    MOV     @RS43,-(SP)
9370 045716 000002          RTI
9371 045720 112737 000200 001764 1$:  MOVB   #200,@RSHSTAT   ;TRY AGAIN
9372 045726 000002          RTI          ;SET DONE FLAG
9373                                     ;RETURN
9374                                     ;*****
9375 :SBTTL UNIBUS EXERCISER SERVICE ROUTINE
9376 :* SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
9377 :*****
9378 UBESRV: SAVREG
9379 JSR    PC,@LDKT          ;GO TO LOW CORE
9380 MOV    @UBETBL+6,R4     ;GET ADDRESS OF UBECR1
9381 TST    @R4              ;WAS THERE AN ERROR?
9382 BMI   UBE2             ;BRANCH IF YES
9383 ADD    #776,@UBESAV     ;INCREMENT UBE BUS ADR
9384 ADC   @UBESAV+2
9385 MOV   @UBESAV,@UBEADR  ;LOAD UBECR2
9386 MOV   @UBEADR+2,@-(R4) ;LOAD UBEBR
9387 MOV   @UBEADR,@-(R4)   ;LOAD UBECR
9388 MOV   #170000,@-(R4)   ;LOAD UBEC
9389 JSR   PC,@RESKT        ;GO BACK TO ORIGINAL CORE
9390 RESREG
9391 MOV   #64545,@UBETBL+6 ;RESTART UBE
9392 RTI          ;RETURN
9393
9394 :UBE ERROR-IS IT LAST MEMORY?
9395 UBE2: CLR   @MEMFLG
9396 SUB   #4,R4             ;ADJUST R4
9397 MOV   @2(R4),R3        ;GET BECR2
9398 BIC   #3,R3            ;GET RID OF ADDRESS BITS
9399 CMP   #400,R3         ;WAS ERROR A TIMEOUT?
9400 BNE   UBEERR          ;BRANCH IF NO
9401 MOV   @R4,@PA1500     ;SAVE BUS ADR OF ERROR
9402 MOV   @2(R4),@PA1716
9403 BIC   #177774,@PA1716
9404 SUB   #4,@PA1500
9405 SBC   @PA1716
9406 CMP   @PA1500,@MXMML0 ;ADJUST PHYSICAL ADR THAT FAILED
9407 BNE   MHOLE          ;UBE STOPS AT ADR+4
9408 CMP   @PA1716,@MXMMHI ;AT MAXIMUM MEMORY LO?
9409 BNE   MHOLE          ;BRANCH IF NO
9410 JSR   PC,@UBEINIT
9411 JSR   PC,@RESKT
9412 RESREG
9413 MOV   #64545,@UBETBL+6 ;AT MAX MEMORY HI?
9414 RTI          ;BRANCH IF NO
9415

```

```

9416 046154 010637 001732      MHOLE:  MOV      SP, @MEMFLG
9417 046160 013737 001212 001734  UBEERR:  MOV      @SLPERR, @ERRADR; SAVE LOOP ERROR ADR
9418 046166 012737 046230 001212      MOV      @UBE3, @SLPERR      ; SET LOOP ADR
9419 046174 012703 000022      MOV      @22, R3
9420 046200 005737 001732      TST      @MEMFLG
9421 046204 001002      BNE      1$
9422 046206 104007      ERROR   7
9423 046210 000407      BR       UBE3
9424 046212 013737 001540 001226 1$:      MOV      @PA1500, @SGDDAT
9425 046220 013737 001542 001230      MOV      @PA1716, @SBDDAT
9426 046226 104012      ERROR   12
9427
9428      ; RESTART UBE IN SAME MEMORY
9429 046230 013737 001734 001212  UBE3:  MOV      @ERRADR, @SLPERR      ; RESTORE ERROR LOOP ADR
9430 046236 010446      MOV      R4, -(SP)           ; SAVE R4
9431 046240 012704 002304      MOV      @UBETBL, R4         ; GET ADDRESS OF UBE TABLE
9432 046244 012734 170000      MOV      @170000, @R4)+     ; SET UBECC
9433 046250 013734 001722      MOV      @UBEADR, @R4)+    ; SET UBEBA <15:00>
9434 046254 005074 000004      CLR      @R4                ; CLEAR ALL ERRORS
9435 046260 013734 001724      MOV      @UBEADR+2, @R4)+   ; SET EXT ADR BITS
9436 046264 012774 064545 000000      MOV      @64545, @R4       ; START UBE
9437 046272 012604      MOV      (SP)+, R4          ; RESTORE R4
9438 046274 004737 054604      JSR      PC, @RESKT
9439 046300 104407      RESREG
9440 046302 000002      RTI                          ; RETURN
9441
9442      ; *****
9443      ; SBTTL MASS BUS TESTER SERVICE ROUTINE
9444      ; * SEE DOCUMENTATION FOR FUNCTIONAL DESCRIPTION OF ROUTINE
9445      ; *****
9446 046304 104406      MBTSRV: SAVREG
9447 046306 004737 054516      JSR      PC, @LDKT          ; GO TO LOW CORE
9448 046312 005037 001732      CLR      @MEMFLG
9449 046316 012704 002322      MOV      @MBTTBL, R4       ; GET ADDRESS OF ADDRESS OF CS1 REG
9450 046322 032734 040000      BIT      @BIT14, @R4)+    ; ANY ERRORS?
9451 046326 001007      BNE      1$                ; BRANCH IF YES
9452 046330 004737 054604 2$:      JSR      PC, @RESKT        ; GO BACK TO ORIGINAL CORE
9453 046334 104407      RESREG
9454 046336 112777 000161 133756      MOVB    @161, @MBTTBL
9455 046344 000002      RTI                          ; RESTART MBT AND RETURN
9456 046346 062704 000010 1$:      ADD      @10, R4           ; ADJUST R4
9457 046352 032774 004000 000000      BIT      @BIT11, @R4      ; NON-EXISTANT MEMORY ERROR?
9458 046360 001443      BEQ      MBTERR            ; BRANCH IF NO
9459 046362 162704 000006      SUB      @6, R4            ; ADJUST R4
9460 046366 013437 001540      MOV      @R4)+, @PA1500    ; GET BUS ADR
9461 046372 013405      MOV      @R4)+, R5        ; GET BUS ADR EXT
9462 046374 072527 000010      ASH     @10, R5           ; SHIFT EXTENDED BITS RIGHT
9463 046400 042705 177774      BIC     @177774, R5       ; CLEAR ALL BITS BUT 0 & 1
9464 046404 010537 001542      MOV      R5, @PA1716
9465 046410 162737 000004 001540      SUB      @4, @PA1500
9466 046416 005637 001542      SBC     @PA1716
9467 046422 023737 001540 001624      CMP     @PA1500, @MXMML0   ; IS IT LAST MEMORY?
9468 046430 001015      BNE     MEMHOLE           ; BRANCH IF NO
9469 046432 023737 001542 001622      CMP     @PA1716, @MXMMHI  ; CHECK EXT ADR BITS
9470 046440 001011      BNE     MEMHOLE
9471 046442 005724      TST     (R4)+             ; INCREMENT R4

```



# E16

```

9472 046444 052774 000047 000000      BIS      #47,2(R4)      ;CLEAR THE ERROR
9473 046452 012734 000007      MOV      #7,2(R4)+ ;SELECT UNIT 7
9474 046456 005074 177766      CLR      2-12(R4) ;CLEAR WORD COUNT
9475 046462 000722      BR       25       ;CONTINUE
9476
9477 046464 010637 001732      MEMHOLE:MOV SP,2#MEMFLG
9478 046470 013737 001212 001734 MBTERR:MOV 2#SLPERR,2#ERRADR ;SAVE LOOP ADDRESS
9479 046476 012737 046540 001212      MOV      #15,2#SLPERR ;SET NEW LOOP ADR
9480 046504 012703 000020      MOV      #20,R3     ;PUT DEVICE ID IN R3
9481 046510 005737 001732      TST      2#MEMFLG
9482 046514 001002      BNE      25
9483 046516 104007      ERROR   7
9484 046520 000407      BR       15
9485 046522 013737 001540 001226 25:      MOV      2#PA1500,2#SGDDAT
9486 046530 013737 001542 001230      MOV      2#PA1716,2#SBDDAT
9487 046536 104013      ERROR   13
9488 046540 013737 001734 001212 15:      MOV      2#ERRADR,2#SLPERR ;RESTORE LOOP ADR
9489 046546 012704 002332      MOV      #MBTTBL+10,R4 ;GET ADR OF MBTTBL+10
9490 046552 015400      MOV      2-(R4),R0    ;GET BUS ADR EXTENDED
9491 046554 015401      MOV      2-(R4),R1    ;GET BUS ADR
9492 046556 015402      MOV      2-(R4),R2    ;GET WORD COUNT
9493 046560 006302      ASL      R2           ;ADJUST WORD COUNT
9494 046562 160201      SUB      R2,R1        ;FORM START ADR OF THIS XFER
9495 046564 005600      SBC      R0
9496 046566 052774 000047 000010      BIS      #47,210(R4) ;CLEAR THE WORLD
9497 046574 012774 000007 000010      MOV      #7,210(R4) ;SELECT UNIT 7
9498 046602 005724      TST      (R4)+        ;ADJUST R4
9499 046604 010134      MOV      R1,2(R4)+    ;RESTORE BUS ADR
9500 046606 010074 000000      MOV      R0,2(R4)
9501 046612 004737 054604      JSR      PC,2#RESKT   ;GO BACK TO ORIGINAL CORE
9502 046616 104407      RESREG
9503 046620 112777 000161 133474      MOVB    #161,2#MBTTBL ;START MBT AGAIN
9504 046626 000002      RTI              ;RETURN
9505
9506 ;*****
9507 ;SBTTL LINE CLOCK SERVICE ROUTINE
9508 ;* THIS ROUTINE FIRST REMAPS PROGRAM EXECUTION TO LOW
9509 ;* MEMORY. IT THEN INCREMENTS AND KEEPS TRACK OF THE
9510 ;* SECOND AND MINUTE COUNTS KEPT IN LOCATIONS "LTICKS"
9511 ;* AND "MTICKS" RESPECTIVELY.
9512 ;*****
9512 046630 104406      LKSRV: SAVREG
9513 046632 004737 054516      JSR      PC,2#LDKT   ;GO TO LOW CORE
9514 046636 105237 001644      INCB    2#LTICKS     ;INCREMENT TICK COUNT
9515 046642 122737 000074 001644      CMPB    #60.,2#LTICKS ;ONE SECOND YET?
9516 046650 001014      BNE     15          ;BRANCH IF NO
9517 046652 105237 001645      INCB    2#LTICKS+1  ;INCREMENT SECOND COUNT
9518 046656 105037 001644      CLR     2#LTICKS    ;CLEAR SECOND COUNT
9519 046662 122737 000074 001645      CMFB    #60.,2#LTICKS+1 ;ONE MINUTE YET?
9520 046670 001004      BNE     15          ;BRANCH IF NO
9521 046672 105037 001645      CLFB    2#LTICKS+1
9522 046676 005237 001642      INC     2#MTICKS    ;INCREMENT MINUTE COUNT
9523 046702 004737 054604      JSR     PC,2#RESKT  ;RESTORE THE KT
9524 046706 104407      RESREG
9525 046710 012737 000100 177546      MOV     #BIT6,2#LKS ;CLEAR READY BIT IN CLOCK
9526 046716 000002      RTI              ;RETURN
9527
  
```

```

9528 .SBTTL SCOPE HANDLER ROUTINE
9529
9530 ::*****
9531 ::*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
9532 ::*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
9533 ::*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
9534 ::*SW14=1 LOOP ON TEST
9535 ::*SW11=1 INHIBIT ITERATIONS
9536 ::*SW09=1 LOOP ON ERROR
9537 ::*CALL
9538 ::* SCOPE ;;SCOPE=IOT
9539
9540 046720 $SCOPE:
9541 046720 032777 040000 132314 1$: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
9542 046726 001077 BNE $OVER ;;YES IF SW14=1
9543 :*****START OF CODE FOR THE XOR TESTER*****
9544 046730 000416 $XTSTR: BR 6$ ;; IF RUNNING ON THE "XOR" TESTER CHANGE
9545 ;; THIS INSTRUCTION TO A "NOP" (NOP=240)
9546 046732 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
9547 046736 012737 046756 000004 MOV #5,$#ERRVEC ;;SET FOR TIMEOUT
9548 046744 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
9549 046750 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
9550 046754 000453 BR $SVLAD ;;GO TO THE NEXT TEST
9551 046756 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
9552 046760 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
9553 046764 000413 BR 7$ ;;LOOP ON THE PRESENT TEST
9554 046766 6$;:*****END OF CODE FOR THE XOR TESTER*****
9555 046766 105767 132212 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
9556 046772 001421 BEQ 3$ ;;BR IF NO
9557 046774 126767 132217 132202 CMFB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
9558 047002 101015 BHI 3$ ;;BR IF NO
9559 047004 032777 001000 132230 BIT #BIT09,$SWR ;;LOOP ON ERROR?
9560 047012 001404 BEQ 4$ ;;BR IF NO
9561 047014 016767 132172 132166 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
9562 047022 000441 BR $OVER
9563 047024 105067 132154 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
9564 047030 005067 132270 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
9565 047034 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
9566 047036 032777 004000 132176 3$: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
9567 047044 001011 BNE 1$ ;;BR IF YES
9568 047046 005767 132126 TST $PASS ;;IF FIRST PASS OF PROGRAM
9569 047052 001406 BEQ 1$ ;;INHIBIT ITERATIONS
9570 047054 005267 132126 INC $ICNT ;;INCREMENT ITERATION COUNT
9571 047060 026767 132240 132120 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
9572 047066 002017 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
9573 047070 012767 000001 132110 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
9574 047076 016767 000054 132220 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
9575 047104 $SVLAD:
9576 047104 011667 132100 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
9577 047110 011667 132076 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
9578 047114 005067 132206 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
9579 047120 112767 000001 132071 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
9580 047126 105767 132052 $OVER: TSTB $ERFLG ;;ANY ERRORS?
9581 047132 001403 BEQ 1$ ;;BRANCH IF NO
9582 047134 116767 132044 132041 MOVB $ERFLG,$TSTNM+1
9583 047142 016777 132034 132074 1$: MOV $TSTNM,$DISPLAY ;;DISPLAY TEST NUMBER
    
```



```

9584 047150 016716 132034      MOV      SLPADR, (SP)      ;; FUDGE RETURN ADDRESS
9585 047154 000002              RTI                      ;; FIXES PS
9586 047156 000010      SMXCNT: 10              ;; MAX. NUMBER OF ITERATIONS
9587                      .SBTTL  ERROR HANDLER ROUTINE

```

```

9588
9589      ;; *****
9590      ;; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
9591      ;; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
9592      ;; AND GO TO SERRTYP ON ERROR
9593      ;; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
9594      ;; SW15=1      HALT ON ERROR
9595      ;; SW13=1      INHIBIT ERROR TYPEOUTS
9596      ;; SW10=1      BELL ON ERROR
9597      ;; SW09=1      LOOP ON ERROR
9598      ;; CALL
9599      ;; *      ERROR      N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
9600

```

```

9601 047160 116737 132020 001203  SERROR:
9602 047160 105267 132012 7$:      MOV      SERFLG, @STSTNM+1
9603 047166 001775              INCB     SERFLG          ;; SET THE ERROR FLAG
9604 047172 016777 132002 132042  BEQ      7$              ;; DON'T LET THE FLAG GO TO ZERO
9605 047174 032777 002000 132032  MOV      STSTNM, @DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
9606 047202 001402              BIT      #BIT10, @SWR     ;; BELL ON ERROR?
9607 047210 104401 001330              BEQ      1$              ;; NO - SKIP
9608 047212 005267 131772              TYPE    $BELL           ;; RING BELL
9609 047216 011667 131772 1$:      INC      $ERTTL         ;; COUNT THE NUMBER OF ERRORS
9610 047222 162767 000002 131764  MOV      (SP), $ERRPC     ;; GET ADDRESS OF ERROR INSTRUCTION
9611 047226 117767 131760 131754  SUB      #2, $ERRPC
9612 047234 032777 020000 131772  MOV      @SERRPC, $ITEMB  ;; STRIP AND SAVE THE ERROR ITEM CODE
9613 047242 001004              BIT      #BIT13, @SWR     ;; SKIP TYPEOUT IF SET
9614 047250 000056              BNE     20$             ;; SKIP TYPEOUTS
9615 047252 104401 001335              JSR     PC, SERRTYP      ;; GO TO USER ERROR ROUTINE
9616 047256 001335              TYPE    , $CRLF
9617 047262 005777 131754 20$:
9618 047262 100001              TST     @SWR             ;; HALT ON ERROR
9619 047266 000000              BPL     3$              ;; SKIP IF CONTINUE
9620 047270 001000 131742 3$:      HALT
9621 047272 001402              BIT     #BIT09, @SWR     ;; HALT ON ERROR!
9622 047300 016716 131704              BEQ     4$              ;; LOOP ON ERROR SWITCH SET?
9623 047302 005767 132014              MOV     SLPERR, (SP)    ;; BR IF NO
9624 047306 001402              TST     $ESCAPE         ;; FUDGE RETURN FOR LOOPING
9625 047312 016716 132006              BEQ     5$              ;; CHECK FOR AN ESCAPE ADDRESS
9626 047314 022737 041230 000042  MOV     $ESCAPE, (SP)   ;; BR IF NONE
9627 047320 001001              CMP     #SENDAD, @#42   ;; FUDGE RETURN ADDRESS FOR ESCAPE
9628 047326 000000              BNE     6$              ;; ACT-11 AUTO-ACCEPT?
9629 047330 000002              HALT                    ;; BRANCH IF NO
9630 047332 000002              RTI                      ;; YES
9631 047332 000002
9632

```

```

9633      ;; *****
9634      .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
9635
9636      ;; THIS ROUTINE FIRST TYPES A STANDARD MESSAGE CONSISTING OF THE
9637      ;; VIRTUAL PC, THE PHYSICAL PC, THE PSW AT THE TIME OF THE ERROR CALL,
9638      ;; AND THE SUB-PASS COUNT. THE SUB-PASS COUNT CONSISTS OF THE SUB PASS COUNT IN THE
9639      ;; HIGH BYTE AND THE PASS COUNT IN THE LOW BYTE.

```

9640  
9641  
9642  
9643  
9644  
9645  
9646  
9647  
9648  
9649  
9650  
9651  
9652  
9653  
9654  
9655  
9656  
9657  
9658  
9659  
9660  
9661  
9662  
9663  
9664  
9665  
9666  
9667  
9668  
9669  
9670  
9671  
9672  
9673  
9674  
9675  
9676  
9677  
9678  
9679  
9680  
9681  
9682  
9683  
9684  
9685  
9686  
9687  
9688  
9689  
9690  
9691  
9692  
9693  
9694  
9695

047334 104406  
047336 104401 001335  
047342 004737 051010  
047346 104401 055306  
047352 104401 001335  
047356 016746 131636  
  
047362 104402  
047364 104401 050556  
047370 013700 001536  
047374 013737 001220 001536  
047402 122737 000014 001216  
047410 003403  
047412 105737 001216  
047416 001005  
047420 004737 053724  
047424 010037 001536  
047430 000407  
047432 013737 001536 001540  
047440 005037 001542  
047444 010037 001536  
047450 012746 001540  
047454 004737 052060  
047460 062716 000005  
047464 012667 000002  
047470 104401  
047472 000000  
047474 104401 050556  
047500 016646 000030  
047504 104402  
047506 104401 050556  
047512 116746 131464  
047516 105066 000001  
047522 104402

```
;*
;*IT THEN USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE"
;*THE ERROR MESSAGE POINTER AND TYPES THE ERROR MESSAGE. THE DATA
;*HEADER POINTER IS THEN OBTAINED AND A DATA HEADER IS TYPED.
;*THE DATA POINTER AND DATA FORMAT ARE THEN OBTAINED. THERE ARE
;*FOUR TYPES OF DATA FORMAT, AS FOLLOWS:
;*
;*      0      TYPE THE CONTENTS OF THE DATA TABLE WORD IN
;*              6 DIGIT OCTAL FORMAT
;*      1      CONVERT THE CONTENTS OF THE DATA TABLE WORD TO
;*              18 BITS AND TYPE AN 6 DIGIT OCTAL NUMBER
;*      2      TYPE THE CONTENTS OF THE DATA TABLE WORD AND
;*              THE WORD+2 IN 6 DIGIT OCTAL FORMAT
;*      3      USE THE CONTENTS OF THE DATA TABLE WORD AS A
;*              DEVICE ID AND TYPE THE DEVICES NAME
;*      4      CONVERT THE TWO WORDS POINTED TO BY THE DATA
;*              TABLE TO FLOATING POINT FORMAT AND TYPE.
;*      5      CONVERT THE FOUR WORDS POINTED TO BY THE DATA
;*              TABLE TO FLOATING DOUBLE FORMAT AND TYPE.
;*****
```

```
SERRTYP: SAVREG
TYPE      $SCLF      ;:"CARRIAGE RETURN" & "LINE FEED"
JSR      PC, $TYPTIME ;GO TYPE THE TIME
TYPE      ,MSG3
TYPE      $SCLF
MOV      $ERRPC, -(SP) ;SAVE $ERRPC FOR TYPEOUT
;TYPE THE VIRTUAL PC
;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYP0C
TYPE      $B
MOV      $VADR, R0 ;SAVE VADR
MOV      $SERRPC, $VADR ;SAVE THE VIR PC FOR CONVERSION
CMPB     #14, $ITEMB
BLE      51$
TSTB     $ITEMB ;ERROR ZERO?
BNE      42$ ;BRANCH IF NO
51$: JSR      PC, $CNVADR ;CONVERT TO 18 BITS
MOV      R0, $VADR
BR       41$
42$: MOV      $VADR, $PA1500
CLR      $PA1716
MOV      R0, $VADR
41$: MOV      $PA1500, -(SP) ;PUT ADDRESS OFPC ON STACK
JSR      PC, $SDB20 ;CONVERT TO ASCII
ADD      #5, (SP) ;GET RID OF 5 MS DIGITS
MOV      (SP)+, 30$ ;SAVE POINTER TO ASCII
TYPE      IT
30$: .WORD
TYPE      $B
MOV      30(SP), -(SP) ;GET PSW AT TIME OF ERROR
TYP0C
TYPE      $B
MOV      $TSTNM, -(SP) ;TYPE IT
CLRB     1(SP)
TYP0C ;TYPE THE TEST NUMBER
```



```

9696 047524 104401 050556      TYPE      8$
9697 047530 013746 001574      MOV      2$SUBPASS,-(SP)
9698 047534 162716 000060      SUB      #60,(SP)
9699 047540 104402      TYPOC
9700 047542 104401 050556      TYPE      8$
9701 047546 016746 131426      MOV      $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
9702                                ;;TYPE THE PASS COUNT
9703 047552 104405      TYPOS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
9704 047554 104401 001335      TYPE      $CRLF
9705 047560 005000      CLR      R0
9706 047562 153700 001216      BISB     2$SITEMB,R0      ;PICK UP THE INDEX
9707 047566 001431      BEQ      6$              ;EXIT IF ZERO
9708 047570 022700 000007      1$: CMP      #7,R0          ;IS THIS ERROR 7?
9709 047574 001551      BEQ      15$            ;BRANCH IF YES
9710 047576 005300      DEC      R0              ;ADJUST THE INDEX SO THAT IT WILL
9711 047600 006300      ASL      R0              ;WORK FOR THE ERROR TABLE
9712 047602 006300      ASL      R0
9713 047604 006300      ASL      R0
9714 047606 062700 003126      ADD      #SERRTB,R0      ;;FORM TABLE POINTER
9715 047612 012067 000004      MOV      (R0)+,2$      ;;PICKUP "ERROR MESSAGE" POINTER
9716 047616 001404      BEQ      3$              ;SKIP TYPEOUT IF NO POINTER
9717 047620 104401      TYPE                                ;;TYPE THE "ERROR MESSAGE"
9718 047622 000000      2$: .WORD 0              ;"ERROR MESSAGE" POINTER GOES HERE
9719 047624 104401 001335      TYPE      $CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
9720 047630 012067 000004      3$: MOV      (R0)+,4$      ;PICKUP "DATA HEADER" POINTER
9721 047634 001404      BEQ      5$              ;SKIP TYPEOUT IF 0
9722 047636 104401      TYPE                                ;;TYPE THE "DATA HEADER"
9723 047640 000000      4$: .WORD 0              ;"DATA HEADER" POINTER GOES HERE
9724 047642 104401 001335      TYPE      $CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
9725 047646 012001 001335      5$: MOV      (R0)+,R1      ;PICKUP "DATA TABLE" POINTER
9726 047650 001004      BNE      7$              ;;GO TYPE THE DATA
9727 047652 104407      RESREG
9728 047654 104401 001335      TYPE      $CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
9729 047660 000207      RTS      PC              ;RETURN
9730 047662 011002      7$: MOV      (R0),R2      ;GET "DATA FORMAT" POINTER
9731 047664 122712 000001      10$: CMPB     #1,(R2)      ;DATA FORMAT 1?
9732 047670 001424      BEQ      9$              ;BRANCH IF YES
9733 047672 122712 000002      CMPB     #2,(R2)      ;DATA FORMAT 2?
9734 047676 001441      BEQ      11$            ;BRANCH IF YES
9735 047700 122712 000003      CMPB     #3,(R2)      ;DATA FORMAT 3?
9736 047704 001445      BEQ      24$            ;BRANCH IF YES
9737 047706 122712 000004      CMPB     #4,(R2)      ;DATA FORMAT 4?
9738 047712 001456      BEQ      40$            ;BRANCH IF YES
9739 047714 122712 000005      CMPB     #5,(R2)      ;DATA FORMAT 5?
9740 047720 001465      BEQ      60$            ;BRANCH IF YES
9741                                ;;*****
9742                                ;DATA FORMAT 0
9743 047722 005202      INC      R2              ;INCREMENT FORMAT POINTER
9744 047724 013146      MOV      2(R1)+,-(SP)    ;PUSH DATA TO BE TYPED
9745 047726 104402      TYPOC
9746 047730 005711      13$: TST      (R1)          ;ANY MORE DATA?
9747 047732 001747      BEQ      1$              ;BRANCH IF NO
9748 047734 104401 050556      TYPE      8$              ;TYPE TWO SPACES
9749 047740 000751      BR       10$
9750                                ;;*****
9751                                ;DATA FORMAT 1

```

9752 047742 005202  
9753 047744 004737 053724  
9754 047750 012746 001540  
9755 047754 004737 052060  
9756 047760 062716 000005  
9757 047764 012667 000002  
9758 047770 104401  
9759 047772 000000  
9760 047774 062701 000002  
9761 050000 000753  
9762  
9763  
9764 050002 005202  
9765 050004 011100  
9766 050006 012037 001540  
9767 050012 011037 001542  
9768 050016 000754  
9769  
9770  
9771 050020 005202  
9772 050022 013167 000016 000010  
9773 050026 062767 055236 000010  
9774 050034 017767 000004 000002  
9775 050042 104401  
9776 050044 000000  
9777 050046 000730  
9778  
9779  
9780 050050 005202  
9781 050052 012167 000002  
9782 050056 104410  
9783 050060 000000  
9784 050062 012667 000002  
9785 050066 104401  
9786 050070 000000  
9787 050072 000716  
9788  
9789  
9790 050074 005202  
9791 050076 012167 000002  
9792 050102 104411  
9793 050104 000000  
9794 050106 012667 000002  
9795 050112 104401  
9796 050114 000000  
9797 050116 000704  
9798  
9799  
9800 050120 010300  
9801 050122 062700 055236  
9802 050126 011067 000002  
9803 050132 104401  
9804 050134 000000  
9805 050136 104401 050144  
9806 050142 000404  
9807

```
9S: INC R2 ; INCREMENT FORMAT POINTER
     JSR PC, @CNVADR ; GET 18 BIT ADR
14S: MOV #PA1500, -(SP) ; PUSH ADR OF 18 BIT ADR
     JSR PC, @SDB20 ; CONVERT TO ASCII
     ADD #5, (SP) ; DELETE LEADING ZEROS
     MOV (SP)+, 12S ; GET ADR OF ASCII STRING
     TYPE
12S: .WORD
     ADD #2, R1 ; INCREMENT R1
     BR 13S
;*****
; DATA FORMAT 2
11S: INC R2 ; INCREMENT FORMAT POINTER
     MOV (R1), RO
     MOV (RO)+, @PA1500
     MOV (RO), @PA1716
     BR 14S
;*****
; DATA FORMAT 3
24S: INC R2 ; INCREMENT FORMAT POINTER
     MOV @R1+, 25S ; GET DEVICE ID
     ADD #MSGINX, 25S ; FORM ADR OF ASCIZ ADR
     MOV @25S, 25S ; GET ADR OF ASCIZ
     TYPE
25S: .WORD
     BR 13S ; CONTINUE
;*****
; DATA FORMAT 4
40S: INC R2
     MOV (R1)+, 44S ; GET ADDRESS OF DATA
     FL20 ; CONVERT TO FLOATING FORMAT
44S: .WORD
     MOV (SP)+, 45S ; GET ADDRESS OF ASCIZ STRING
     TYPE ; TYPE THE DATA
45S: .WORD
     BR 13S
;*****
; DATA FORMAT 5
60S: INC R2 ; INCREMENT FORMAT POINTER
     MOV (R1)+, 61S ; GET ADDRESS OF DATA
     FLD20 ; CONVERT TO FLOATING ASCII
61S: .WORD
     MOV (SP)+, 62S ; GET ADDRESS OF ASCII STRING
     TYPE ; TYPE THE DATA
62S: .WORD
     BR 13S
;*****
; ERROR 7 DECODE
15S: MOV R3, RO ; SAVE R3
     ADD #MSGINX, RO ; GEN ADRS OF ASCIZ
     MOV (RO), 16S
     TYPE
16S: .WORD
     TYPE 65S ; TYPE ASCIZ STRING
     BR 64S ; GET OVER THE ASCIZ
; 65S: .ASCIZ /FAILED/<CRLF>
```



# K16

```

9808 050154
9809 050154 010300
9810 050156 022700 000010
9811 050162 003403
9812 050164 104401 055533
9813 050170 000411
9814
9815
9816 050172 022703 000020
9817 050176 001426
9818 050200 002435
9819 050202 104401 055646
9820 050206 022700 000012
9821 050212 001131
9822
9823
9824 050214 062700 055212
9825 050220 011000
9826 050222 022703 000002
9827 050226 001404
9828 050230 100406
9829 050232 012704 000007
9830 050236 000423
9831 050240 012704 000006
9832 050244 000420
9833 050246 012704 000011
9834 050252 000415
9835 050254 104401 056035
9836 050260 012704 000011
9837 050264 062700 055212
9838 050270 011000
9839 050272 000405
9840 050274 104401 056144
9841 050300 012704 000004
9842 050304 000767
9843 050306 013046
9844 050310 104402
9845 050312 104401 050556
9846 050316 077405
9847
9848
9849 050320 022703 000022
9850 050324 001452
9851 050326 022703 000002
9852 050332 002443
9853 050334 001005
9854 050336 104401 056340
9855 050342 012700 002202
9856 050346 000404
9857 050350 012700 002160
9858 050354 104401 056350
9859 050360 013001
9860 050362 005720
9861 050364 013037 001540
9862 050370 072127 177774
9863 050374 042701 177774

64$:
MOV R3,RO ;SAVE DEVICE ID
CMP #10,RO ;MASS BUS DEVICE?
BLE 17$ ;BRANCH IF YES
TYPE MSG12
BR 18$

;*****
;MASS BUS ERR
17$:
CMP #20,R3 ;MBT ERROR?
BEQ 26$ ;BRANCH IF MBT ERROR
BLT 27$ ;BRANCH IF UBE ERROR
TYPE MSG13
CMP #12,RO ;WAS IT RS?
BNE 29$ ;BRANCH IF NO

;*****
;UNIBUS ERROR OR RSO4 ERROR
18$:
ADD #REGINX,RO ;FORM ADR OF REG TABLE
MOV (RO),RO ;GET ADR OF REG TABLE
CMP #2,R3 ;RP3 OR RK?
BEQ 20$ ;BRANCH IF RK
BMI 21$ ;BRANCH IF NOT RP03
MOV #7,R4 ;SET RP03 SOB COUNT
BR 22$
20$:
MOV #6,R4 ;SET RK05 SOB COUNT
BR 22$
21$:
MOV #11,R4 ;SET RSO4 SOB COUNT
BR 22$
26$:
TYPE MSG16 ;SET MBT SOB COUNT
MOV #11,R4
28$:
ADD #REGINX,RO ;GET ADR OF MBT TABLE
MOV (RO),RO ;GO TYPE REGISTERS
BR 22$
27$:
TYPE MSG17 ;SET UBE SOB COUNT
MOV #4,R4 ;GO TYPE UBE REGISTERS
BR 28$
22$:
MOV 2(RO)+,-(SP) ;GET DATA IN REG
TYPOC ;TYPE IT
TYPE 8$ ;TYPE TWO SPACES
SOB R4,22$ ;CONTINUE

;*****
;THIS CODE TYPES A PHYSICAL BUS ADDRESS IF THE ERROR WAS AN RP03 OR RK05
CMP #22,R3 ;UBE ERROR?
BEQ 73$ ;BRANCH IF YES
CMP #2,R3 ;RK05?
BLT 32$ ;BRANCH IF NOT RK OR RP03
BNE 70$ ;BRANCH IF RP03
TYPE MSG22
MOV #RKCS,RO ;GET ADR OF ADR OF RKCS REG
BR 71$
70$:
MOV #RP3CS,RO ;GET ADR OF ADR OF RP3CS REG
TYPE MSG23
71$:
MOV 2(RO)+,R1 ;GET BUS ADR EXTENDED BITS
TST (RO)+ ;ADJUST RO
MOV 2(RO)+,2#PA1500 ;GET BUS ADDRESS THAT FAILED
ASH #4,R1 ;GET BITS 4&5 INTO BITS 0&1
BIC #177774,R1 ;GET RID OF UNUSED BITS
  
```

```

9864 050400 010137 001542      MOV      R1,2#PA1716      ;SAVE EXTENDED BITS
9865 050404 162737 000002 001540 74$: SUB      #2,2#PA1500      ;DECREMENT BUS ADR
9866 050412 005637 001542      SBC      2#PA1716
9867 050416 012746 001540      MOV      #PA1500, -(SP)
9868 050422 004737 052060      JSR      PC,2#SOB20      ;CONVERT TO ASCIZ STRING
9869 050426 062716 000003      ADD      #3, (SP)        ;GET RID OF LEADING ZEROS
9870 050432 012667 000002      MOV      (SP)+, 72$
9871 050436 104401
9872 050440 000000      72$: .WORD
9873 050442 104401 001335      32$: .TYPE      ,SCLF
9874 050446 000167 177200      JMP      6$
9875 050452 012700 002306      73$: MOV      #UBETBL+2, R0      ;GET ADR OF UBE TABLE +2
9876 050456 013037 001540      MOV      2(R0)+, 2#PA1500 ;GET BUS ADR THAT FAILED
9877 050462 013037 001542      MOV      2(R0)+, 2#PA1716 ;GET BAE BITS
9878 050466 042737 177774 001542      BIC      #177774, 2#PA1716; MASK OFF ADR BITS
9879 050474 000743      BR      74$
9880
9881      ;*****
9882      ;RPO4 ERROR
9883 050476 062700 055212      29$: ADD      #REGINX, R0
9884 050502 011000      MOV      (R0) R0      ;FORM ADR OF RPG4 TABLE
9885 050504 012704 000011      MOV      #11, R4      ;SET SOB COUNT
9886 050510 013046      MOV      2(R0)+, -(SP)   ;GET DATA TO BE TYPED
9887 050514 104401 050556      TYPOC   ;TYPE DATA
9888 050520 077405      TYPE    8$
9889 050522 104401 001335      SOB     R4, 31$        ;CONTINUE
9890 050526 104401 001335      TYPE    ,SCLF
9891 050532 012704 000004      MOV     #4, R4        ;SET SOB COUNT
9892 050536 104401 055756      TYPE    MSG14
9893 050542 013046      50$: MOV     2(R0)+, -(SP) ;GET DTA TO BE TYPED
9894 050544 104402      TYPOC   ;TYPE IT
9895 050546 104401 050556      TYPE    8$
9896 050552 077405      SOB     R4, 50$      ;CONTINUE
9897 050554 000732      BR      32$
9898 050556 020040 000      8$: .ASCIZ  / /        ;; TWO(2) SPACES
9899      .EVEN
9900      .SBTTL TYPE ROUTINE
9901
9902      ;*****
9903      ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
9904      ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
9905      ;NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
9906      ;NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
9907      ;NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
9908      ;
9909      ;CALL:
9910      ;1) USING A TRAP INSTRUCTION
9911      ;      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
9912      ;OR
9913      ;      TYPE
9914      ;      MESADR
9915      ;
9916
9917 050562 105767 130473      $TYPE: TSTB   $TPFLG      ;; IS THERE A TERMINAL?
9918 050566 100002      BPL     1$            ;; BR IF YES
9919 050570 000000      HALT

```



M16

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58 TYPE ROUTINE

MACY11 27(1006) 07-FEB-77 10:08 PAGE 207

```

9920 050572 000407          BR      3$          ;; LEAVE
9921 050574 010046          1$: MOV    RO, -(SP)      ;; SAVE RO
9922 050576 017600 000002  MOV    2(SP), RO      ;; GET ADDRESS OF ASCIZ STRING
9923 050602 112046          2$: MOVB   (RO)+, -(SP)  ;; PUSH CHARACTER TO BE TYPED ONTO STACK
9924 050604 001005          BNE    4$          ;; BR IF IT ISN'T THE TERMINATOR
9925 050606 005726          TST    (SP)+        ;; IF TERMINATOR POP IT OFF THE STACK
9926 050610 012600          60$: MOV    (SP)+, RO      ;; RESTORE RO
9927 050612 062716 000002  3$: ADD    #2, (SP)     ;; ADJUST RETURN PC
9928 050616 000002          RTI                    ;; RETURN
9929 050620 122716 000011  4$: CMPB   #HT, (SP)    ;; BRANCH IF <HT>
9930 050624 001430          BEQ    8$          ;; BRANCH IF NOT <CRLF>
9931 050626 122716 000200  CMPB   #CRLF, (SP)
9932 050632 001006          BNE    5$          ;; POP <CR><LF> EQUIV
9933 050634 005726          TST    (SP)+        ;; TYPE A CR AND LF
9934 050636 104401          TYPE
9935 050640 001335          $CRLF
9936 050642 105067 000136  CLRB   $CHARCNT      ;; CLEAR CHARACTER COUNT
9937 050646 000755          BR     2$          ;; GET NEXT CHARACTER
9938 050650 004767 000056  5$: JSR    PC, $TYPEC    ;; GO TYPE THIS CHARACTER
9939 050654 126726 130400  6$: CMPB   $FILLC, (SP)+  ;; IS IT TIME FOR FILLER CHARS.?
9940 050660 001350          BNE    2$          ;; IF NO GO GET NEXT CHAR.
9941 050662 016746 130370  MOV    $NULL, -(SP)  ;; GET # OF FILLER CHARS. NEEDED
9942                                AND THE NULL CHAR.
9943 050666 105366 000001  7$: DECB   1(SP)       ;; DOES A NULL NEED TO BE TYPED?
9944 050672 002770          BLT    6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
9945 050674 004767 000032  JSR    PC, $TYPEC    ;; GO TYPE A NULL
9946 050700 105367 000100  DECB   $CHARCNT      ;; DO NOT COUNT AS A COUNT
9947 050704 000770          BR     7$          ;; LOOP
9948
9949                                ; HORIZONTAL TAB PROCESSOR
9950
9951 050706 112716 000040  8$: MOVB   #' (SP)     ;; REPLACE TAB WITH SPACE
9952 050712 004767 000014  9$: JSR    PC, $TYPEC    ;; TYPE A SPACE
9953 050716 132767 000007 000060  BITB   #7, $CHARCNT  ;; BRANCH IF NOT AT
9954 050724 001372          BNE    9$          ;; TAB STOP
9955 050726 005726          TST    (SP)+        ;; POP SPACE OFF STACK
9956 050730 000724          BR     2$          ;; GET NEXT CHARACTER
9957 050732 005737 001530  $TYPEC: TST    $NOTYPE    ;; INHIBIT TYPING?
9958 050736 100423          BMI    $TYPEX      ;; BRANCH IF YES
9959 050740 105777 130306  TSTB   $STPS        ;; WAIT UNTIL PRINTER IS READY
9960 050744 100372          BPL    $TYPEC
9961 050746 116677 000002 130300  MOVB   2(SP), $STPB  ;; LOAD CHAR TO BE TYPED INTO DATA REG.
9962 050754 122766 000015 000002  CMPB   #CR, 2(SP)   ;; IS CHARACTER A CARRIAGE RETURN?
9963 050762 001003          BNE    1$          ;; BRANCH IF NO
9964 050764 105067 000014  CLRB   $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
9965 050770 000406          BR     $TYPEX      ;; EXIT
9966 050772 122766 000012 000002  1$: CMPB   #LF, 2(SP)  ;; IS CHARACTER A LINE FEED?
9967 051000 001402          BEQ    $TYPEX      ;; BRANCH IF YES
9968 051002 105227          INCB   (PC)+        ;; COUNT THE CHARACTER
9969 051004 000000          $CHARCNT: .WORD    0 ;; CHARACTER COUNT STORAGE
9970 051006 000207          $TYPEX: RTS      PC
9971
9972                                ; *****
9973                                ; $BTTL ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM
9974                                ; * THIS ROUTINE CONVERTS THE CONTENTS OF LOCATIONS "LTICK"
9975                                ; * AND "MTICKS" TO SECONDS AND MINUTES/HOURS RESPECTIVELY
    
```

# B01

MAINDEC-11-D0KDC-A PDP 11/6X SERIES CPU EXERCISER  
 D0KDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 208  
 ROUTINE TO TYPE THE ELAPSED RUN TIME OF THE PROGRAM

```

9976
9977
9978
9979 051010 104406
9980 051012 004737 054516
9981 051016 113701 001645
9982 051022 005000
9983 051024 071027 000012
9984 051030 062701 000060
9985 051034 110137 051246
9986 051040 010001
9987 051042 005000
9988 051044 071027 000006
9989 051050 062701 000060
9990 051054 110137 051245
9991 051060 013701 001642
9992 051064 005000
9993 051066 071027 000012
9994 051072 062701 000060
9995 051076 110167 000141
9996 051102 010001
9997 051104 005000
9998 051106 071027 000006
9999 051112 062701 000060
10000 051116 110167 000120
10001 051122 005700
10002 051124 001434
10003 051126 010001
10004 051130 005000
10005 051132 071027 000012
10006 051136 062701 000060
10007 051142 110167 000072
10008 051146 005700
10009 051150 001422
10010 051152 010001
10011 051154 005000
10012 051156 071027 000010
10013 051162 062701 000060
10014 051166 110167 000045
10015 051172 005700
10016 051174 001410
10017 051176 010001
10018 051200 005000
10019 051202 071027 000012
10020 051206 062701 000060
10021 051212 110167 000020
10022 051216 104401 051236
10023 051222 104401 001335
10024 051226 004737 054604
10025 051232 104407
10026 051234 000207
10027 051236 001 001 001 TIMEBUF:.BYTE
10028 051241 072 001 001
10029 051244 072 060 060
10030 051247 000
10031
  
```

```

;* AND TYPES THEM IN THE FOLLOWING FORMAT:
;* MM:MM:SS
;*****
TYPTIME:SAVREG
      JSR   PC,@LDKT      ;GO BACK TO LOW CORE
      MOVB  @LTICKS+1,R1  ;GET SECOND COUNT
      CLR   RO
      DIV  #10.,RO
      ADD  #60,R1
      MOVB  R1,@TIMEBUF+10
      MOV   RO,R1
      CLR   RO
      DIV  #6.,RO
      ADD  #60,R1
      MOVB  R1,@TIMEBUF+7  ;GET MINUTE COUNT
      MOV   @MTICKS,R1
      CLR   RO
      DIV  #10.,RO      ;GET HOURS AND MINUTES
      ADD  #60,R1      ;MAKE REMAINDER ASCII
      MOVB  R1,↑TIMEBUF+5 ;PUT IN BUFFER
      MOV   RO,R1
      CLR   RO
      DIV  #6.,RO
      ADD  #60,R1
      MOVB  R1,↑TIMEBUF+4
      TST  RO
      BEQ  2$
      MOV   RO,R1
      CLR   RO
      DIV  #10.,RO
      ADD  #60,R1
      MOVB  R1,↑TIMEBUF+2
      TST  RO
      BEQ  2$
      MOV   RO,R1
      CLR   RO
      DIV  #10.,RO
      ADD  #60,R1
      MOVB  R1,↑TIMEBUF+1
      TST  RO
      BEQ  2$
      MOV   RO,R1
      CLR   RO
      DIV  #10.,RO
      ADD  #60,R1
      MOVB  R1,↑TIMEBUF
2$:   TYPE  ,↑TIMEBUF
      TYPE  ,SCRLF
      JSR   PC,@RESKT     ;GO BACK TO ORIGINAL MEMORY
      RESREG
      RTS   PC
      TIMEBUF:.BYTE 1,1,1,72,1,1,72,60,60,0
      .EVEN
  
```



CO1

10032  
 10033  
 10034  
 10035  
 10036  
 10037  
 10038  
 10039 051250 104401 057740  
 10040 051254 104401 056401  
 10041 051260 013746 001532  
 10042 051264 104402  
 10043 051266 104401 001335  
 10044 051272 104401 055366  
 10045 051276 012700 000010  
 10046 051302 005001  
 10047 051304 105761 001646  
 10048 051310 001004  
 10049 051312 062701 000002  
 10050 051316 077006  
 10051 051320 000207  
 10052 051322 010102  
 10053 051324 062702 055236  
 10054 051330 011267 000002  
 10055 051334 104401  
 10056 051336 000000  
 10057 051340 112767 000060 000034  
 10058 051346 116102 001646  
 10059 051352 012703 000010  
 10060 051356 006002  
 10061 051360 103002  
 10062 051362 104401 051402  
 10063 051366 005267 000010  
 10064 051372 077307  
 10065 051374 104401 001335  
 10066 051400 000744  
 10067 051402 000 054 040  
 10068 051405 000  
 10069  
 10070  
 10071  
 10072  
 10073  
 10074  
 10075  
 10076  
 10077  
 10078  
 10079  
 10080  
 10081  
 10082  
 10083  
 10084  
 10085  
 10086  
 10087

```

*****
.SBTTL ROUTINE TO TYPE THE AVAILABLE DEVICES AND UNIT NUMBERS
;* THIS ROUTINE SEARCHES THE SYSTEM SIZE TABLE FOR NON-
;* ZERO ENTRIES. WHEN IT FINDS ONE, IT TYPES THE NAME OF THE
;* DEVICE AND THE UNIT NUMBERS THAT WERE FOUND TO BE
;* AVAILABLE FOR THAT DEVICE.
*****
TYPsiz: TYPE ,SWITCH
        TYPE MSG25
        MOV #OPT.CP,-(SP) ;PUT CONTENTS OF OPT.CP ON STACK
        TYPoc ;TYPE OPT.CP
        TYPE ,SCRLF
        TYPE MSG4
        MOV #10,R0 ;SET SOB COUNT
        CLR R1
15: TSTB SYSSize(R1) ;DEVICE AVAILABLE?
        BNE 25 ;BRANCH IF YES
75: ADD #2,R1 ;INCREMENT INDEX
        SOB R0,15 ;CONTINUE
        RTS PC ;RETURN
25: MOV R1,R2 ;GET INDEX
        ADD #MSGINX,R2 ;GET ADR OF MESSAGE ADR
        MOV (R2),3$ ;GET ADDRESS OF MESSAGE
        TYPE
35: .WORD
        MOVb #60,4$ ;INIT UNIT NO. BUFFER (ASCII)
        MOVb SYSSize(R1),R2 ;GET WORD WITH AVAILABLE UNITS
        MOV #10,R3 ;SET SOB COUNT
65: ROR R2 ;GET UNITS
        BCC 55 ;BRANCH IF NOT A UNIT
        TYPE 4$
55: INC 4$
        SOB R3,65 ;CONTINUE
        TYPE ,SCRLF
45: BR 7$
        .BYTE 0,54,40,0 ;NUMBER,COMMA,SPACE,TERMINATOR

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
;* THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;* OCTAL (ASCII) NUMBER AND TYPE IT.
;* $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;* CALL:
;* MOV NUM,-(SP) ;NUMBER TO BE TYPED
;* TYPOS ;CALL FOR TYPEOUT
;* .BYTE N ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;* .BYTE M ;M=1 OR 0
;* ;1=TYPE LEADING ZEROS
;* ;0=SUPPRESS LEADING ZEROS
;* $TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;* $TYPOS OR $TYPOC
;* CALL:
;* MOV NUM,-(SP) ;NUMBER TO BE TYPED
;* TYPON ;CALL FOR TYPEOUT
    
```

# DO1

```

10088
10089
10090
10091
10092
10093
10094 051406 017646 000000
10095 051412 116667 000001 000211
10096 051420 112667 000207
10097 051424 062716 000002
10098 051430 000406
10099 051432 112767 000001 000171
10100 051440 112767 000006 000165
10101 051446 112767 000005 000154
10102 051454 010346
10103 051456 010446
10104 051460 010546
10105 051462 116704 000145
10106 051466 005404
10107 051470 062704 000006
10108 051474 110467 000132
10109 051500 116704 000125
10110 051504 016605 000012
10111 051510 005003
10112 051512 006105
10113 051514 000404
10114 051516 006105
10115 051520 006105
10116 051522 006105
10117 051524 010503
10118 051526 006103
10119 051530 105367 000076
10120 051534 100016
10121 051536 042703 177770
10122 051542 001002
10123 051544 005704
10124 051546 001403
10125 051550 005204
10126 051552 052703 000060
10127 051556 052703 000040
10128 051562 110367 000040
10129 051566 104401 051626
10130 051572 105367 000032
10131 051576 003347
10132 051600 002402
10133 051602 005204
10134 051604 000744
10135 051606 012605
10136 051610 012604
10137 051612 012603
10138 051614 016666 000002 000004
10139 051622 012616
10140 051624 000002
10141 051626 000
10142 051627 000
10143 051630 000

; *
; * $STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
; * CALL:
; * MOV NUM,-(SP) ;: NUMBER TO BE TYPED
; * TYPOC ;: CALL FOR TYPEOUT

STYPOS: MOV 2(SP),-(SP) ;: PICKUP THE MODE
MOV 1(SP),SOFILL ;: LOAD ZERO FILL SWITCH
MOV (SP)+,SOMODE+1 ;: NUMBER OF DIGITS TO TYPE
ADD #2,(SP) ;: ADJUST RETURN ADDRESS
BR STYPOC

STYPOC: MOV #1,SOFILL ;: SET THE ZERO FILL SWITCH
MOV #6,SOMODE+1 ;: SET FOR SIX(6) DIGITS
STYPON: MOV #5,SOCNT ;: SET THE ITERATION COUNT
MOV R3,-(SP) ;: SAVE R3
MOV R4,-(SP) ;: SAVE R4
MOV R5,-(SP) ;: SAVE R5
MOV #SOMODE+1,R4 ;: GET THE NUMBER OF DIGITS TO TYPE
NEG R4
ADD #6,R4 ;: SUBTRACT IT FOR MAX. ALLOWED
MOV R4,SOMODE ;: SAVE IT FOR USE
MOV SOFILL,R4 ;: GET THE ZERO FILL SWITCH
MOV 12(SP),R5 ;: PICKUP THE INPUT NUMBER
CLR R3 ;: CLEAR THE OUTPUT WORD
1S: ROL R5 ;: ROTATE MSB INTO "C"
BR 3S ;: GO DO MSB
2S: ROL R5 ;: FORM THIS DIGIT
ROL R5
MOV R5,R3
3S: ROL R3 ;: GET LSB OF THIS DIGIT
DECB SOMODE ;: TYPE THIS DIGIT?
BPL 7S ;: BR IF NO
BIC #177770,R3 ;: GET RID OF JUNK
BNE 4S ;: TEST FOR 0
TST R4 ;: SUPPRESS THIS 0?
BEQ 5S ;: BR IF YES
4S: INC R4 ;: DON'T SUPPRESS ANYMORE 0'S
BIS #'0,R3 ;: MAKE THIS DIGIT ASCII
5S: BIS #' ,R3 ;: MAKE ASCII IF NOT ALREADY
MOV R3,#S ;: SAVE FOR TYPING
TYPE #S ;: GO TYPE THIS DIGIT
7S: DECB SOCNT ;: COUNT BY 1
BGT 2S ;: BR IF MORE TO DO
BLT 6S ;: BR IF DONE
INC R4 ;: INSURE LAST DIGIT ISN'T A BLANK
BR 2S ;: GO DO THE LAST DIGIT
6S: MOV (SP)+,R5 ;: RESTORE R5
MOV (SP)+,R4 ;: RESTORE R4
MOV (SP)+,R3 ;: RESTORE R3
MOV 2(SP),4(SP) ;: SET THE STACK FOR RETURNING
MOV (SP)+,(SP)

;: RETURN
8S: .BYTE 0 ;: STORAGE FOR ASCII DIGIT
;: TERMINATOR FOR TYPE ROUTINE
SOCNT: .BYTE 0 ;: OCTAL DIGIT COUNTER

```





# FO1

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER      MACY11 27(1006) 07-FEB-77 10:08 PAGE 212  
 DQKDCR.P11      07-FEB-77 09:58      CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

10200 052012 012603          MOV      (SP)+,R3          ;; POP STACK INTO R3
10201 052014 012602          MOV      (SP)+,R2          ;; POP STACK INTO R2
10202 052016 012601          MOV      (SP)+,R1          ;; POP STACK INTO R1
10203 052020 012600          MOV      (SP)+,R0          ;; POP STACK INTO R0
10204 052022 104401 052050  TYPE      $DBLK          ;; NOW TYPE THE NUMBER
10205 052026 016666 000002 000004  MOV      2(SP),4(SP)      ;; ADJUST THE STACK
10206 052034 012616          MOV      (SP)+,(SP)
10207 052036 000002          RTI                          ;; RETURN TO USER
10208 052040 023420          $DTBL: 10000.
10209 052042 001750          1000.
10210 052044 000144          100.
10211 052046 000012          10.
10212 052050 000004          $DBLK: .BLKW 4
10213          .SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
10214
10215          ;; *****
10216          ;; THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
10217          ;; UNSIGNED OCTAL ASCII NUMBER.
10218          ;; CALL
10219          ;; *      MOV      #PNTR, -(SP)          ;; POINTER TO LOW WORD OF BINARY NUMBER
10220          ;; *      JSR      PC, @#$DB20          ;; CALL THE ROUTINE
10221          ;; *      RETURN          ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
10222
10223
10224 052060 104406          $DB20: SAVREG          ;; SAVE ALL REGISTERS
10225 052062 016601 000002  MOV      2(SP),R1          ;; PICKUP THE POINTER TO LOW WORD
10226 052066 012705 052215  MOV      #SOCTVL+13.,R5   ;; POINTER TO DATA TABLE
10227 052072 012704 000014  MOV      #12.,R4          ;; DO ELEVEN CHARACTERS
10228 052076 012703 177770  MOV      #1C7,R3          ;; MASK
10229 052102 012100          MOV      (R1)+,R0          ;; LOWER WORD
10230 052104 012101          MOV      (R1)+,R1          ;; HIGH WORD
10231 052106 005002          CLR      R2                ;; TERMINATOR
10232 052110 110245          15:  MOVVB  R2,-(R5)          ;; PUT CHARACTER IN DATA TABLE
10233 052112 010002          MOV      R0,R2            ;; GET THIS DIGIT
10234 052114 005304          DEC      R4                ;; COUNT THIS CHARACTER
10235 052116 003016          BGT     3$                ;; BR IF NOT THE LAST DIGIT
10236 052120 001414          BEQ     2$                ;; BR IF IT IS THE LAST DIGIT
10237 052122 005205          INC     R5                ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
10238 052124 010566 000002  MOV      R5,2(SP)          ;; ASCII CHAR. & PUT IT ON THE STACK
10239 052130 122765 000061 000003  CMPB   #61,3(R5)          ;; LAST NUMBER LEGAL?
10240 052136 002003          BGE     4$                ;; BRANCH IF YES
10241 052140 112765 000060 000003  MOVVB  #60,3(R5)          ;; MAKE IT A ZERO
10242 052146 104407          4$:  RESREG          ;; RESTORE ALL REGISTERS
10243 052150 000207          RTS     PC                ;; RETURN TO USER
10244 052152 006203          2$:  ASR     R3                ;; POSITION THE MASK FOR THE LAST DIGIT
10245 052154 006001          3$:  ROR     R1                ;; POSITION THE BINARY NUMBER FOR
10246 052156 006000          ROR     R0                ;; THE NEXT OCTAL DIGIT
10247 052160 006001          ROR     R1
10248 052162 006000          ROR     R0
10249 052164 006001          ROR     R1
10250 052166 006000          ROR     R0
10251 052170 040302          BIC     R3,R2            ;; MASK OUT ALL JUNK
10252 052172 062702 000060  ADD     #'0,R2            ;; MAKE THIS CHAR. ASCII
10253 052176 000744          BR      1$                ;; GO PUT IT IN THE DATA TABLE
10254 052200 000016          $OCTVL: .BLKW 14.          ;; RESERVE DATA TABLE
10255          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```



10256  
10257  
10258  
10259  
10260  
10261  
10262  
10263  
10264  
10265  
10266  
10267  
10268  
10269  
10270  
10271  
10272 052216  
10273 052216 010046  
10274 052220 010146  
10275 052222 010246  
10276 052224 010346  
10277 052226 010446  
10278 052230 010546  
10279 052232 016646 000022  
10280 052236 016646 000022  
10281 052242 016646 000022  
10282 052246 016646 000022  
10283 052252 000002  
10284  
10285  
10286  
10287  
10288 052254  
10289 052254 012666 000022  
10290 052260 012666 000022  
10291 052264 012666 000022  
10292 052270 012666 000022  
10293 052274 012605  
10294 052276 012604  
10295 052300 012603  
10296 052302 012602  
10297 052304 012601  
10298 052306 012600  
10299 052310 000002  
10300  
10301  
10302  
10303  
10304  
10305  
10306  
10307  
10308  
10309  
10310  
10311

```
*****  
: *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  
RTI  
  
*****  
: *SBTTL CONVERT FLOATING BINARY TO OCTAL ASCIZ  
: *  
: *THIS ROUTINE CONVERTS A 32 BIT FLOATING NUMBER TO AN OCTAL  
: *ASCIZ STRING IN THE FOLLOWING FORMAT:  
: *  
: * W XXX YYY ZZZZZZ  
: *  
: * WHERE W = SIGN BIT  
: * X = 8-BIT EXPONENT (RIGHT JUSTIFIED)  
: * Y = FRACTION BITS (57:51) (RIGHT JUSTIFIED)  
: * Z = FRACTION BITS (50:35)  
: *
```

# H01

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER  
 DQKDC.A.P11 07-FEB-77 09:58

MACY11 27(1006) 07-FEB-77 10:08 PAGE 214  
 CONVERT FLOATING BINARY TO OCTAL ASCIZ

```

10312
10313
10314
10315
10316
10317 052312 104406
10318 052314 017600 000000
10319 052320 062716 000002
10320 052324 016001 000002
10321 052330 011000
10322 052332 012704 001365
10323 052336 112744 000000
10324 052342 012705 000005
10325 052346 010103
10326 052350 042703 177770
10327 052354 062703 000060
10328 052360 110344
10329 052362 073027 177775
10330 052366 077511
10331 052370 010103
10332 052372 042703 177776
10333 052376 062703 000060
10334 052402 110344
10335 052404 112744 000040
10336 052410 073027 177777
10337 052414 012705 000002
10338 052420 010103
10339 052422 042703 177770
10340 052426 062703 000060
10341 052432 110344
10342 052434 073027 177775
10343 052440 077511
10344 052442 010103
10345 052444 042703 177776
10346 052450 062703 000060
10347 052454 110344
10348 052456 112744 000040
10349 052462 112744 000040
10350 052466 072127 177777
10351 052472 012705 000002
10352 052476 010103
10353 052500 042703 177770
10354 052504 062703 000060
10355 052510 110344
10356 052512 072127 177775
10357 052516 077511
10358 052520 010103
10359 052522 042703 177774
10360 052526 062703 000060
10361 052532 110344
10362 052534 112744 000040
10363 052540 112744 000040
10364 052544 042700 177776
10365 052550 062700 000060
10366 052554 110044
10367 052556 104407
  
```

```

;*
;*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
;*NUMBER IN THE WORD FOLLOWING THE CALL.
;*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
;*****
$FL20: SAVREG
MOV 2(SP),R0 ;GET ADDRESS OF DATA
ADD #2,(SP) ;ADJUST RETURN PC
MOV 2(R0),R1 ;PUT SECOND DATA WORD IN R1
MOV (R0),R0 ;PUT FIRST DATA WORD IN R0
MOV #SFLBUFF+23,R4 ;GET ADDRESS OF BUFFER END IN R4
MOV #0,-(R4) ;PUT TERMINATOR IN BUFFER
MOV #5,R5 ;SET SOB COUNT FOR FRACTION DIGITS
15: MOV R1,R3 ;GET LSB'S OF FRACTION
BIC #17,R3 ;SAVE LS 3 BITS
ADD #60,R3 ;MAKE THEM ASCII
MOV #R3,-(R4) ;STORE IN BUFFER
ASHC #-3,R0 ;SHIFT NUMBER TO NEXT 3 BITS
SOB R5,15 ;CONTINUE FOR 7 DIGITS
MOV R1,R3 ;GET NEXT DIGITS
BIC #1,R3 ;ONLY WANT 1 BIT
ADD #60,R3 ;MAKE THEM ASCII
MOV #R3,-(R4) ;STORE IN BUFFER
MOV #40,-(R4) ;PUT SPACE IN BUFFER
ASHC #-1,R0
35: MOV #2,R5 ;SET SOB COUNT
MOV R1,R3 ;GET LOW WORD
BIC #17,R3 ;MASK 3 BITS
ADD #60,R3 ;MAKE THEM ASCII
MOV #R3,-(R4) ;PUT IN BUFFER
ASHC #-3,R0 ;GET NEXT 3 BITS
SOB R5,35 ;CONVERT THEM
MOV R1,R3
BIC #1,R3 ;ONLY WANT 1 BIT
ADD #60,R3 ;MAKE IT ASCII
MOV #R3,-(R4) ;PUT IN BUFFER
MOV #40,-(R4) ;PUT SPACE IN BUFFER
25: MOV #1,R1 ;GET FIRST 3 BITS OF EXPONENT
MOV #2,R5 ;SET SOB COUNT FOR 2 DIGITS
MOV R1,R3 ;GET LSB'S OF EXPONENT
BIC #17,R3 ;SAVE 3 BITS
ADD #60,R3 ;MAKE THEM ASCII
MOV #R3,-(R4) ;STORE IN BUFFER
ASHC #-3,R1 ;GET NEXT 3 BITS
SOB R5,25 ;CONTINUE
MOV R1,R3 ;GET LAST 2 BITS OF EXPONENT
BIC #3,R3 ;MAKE SURE ONLY 2 BITS
ADD #60,R3 ;MAKE THEM ASCII
MOV #R3,-(R4) ;STORE IN BUFFER
MOV #40,-(R4) ;PUT SPACE IN BUFFER
MOV #40,-(R4)
BIC #1,R0 ;GET SIGN BIT (IT WAS EXTENDED)
ADD #60,R0 ;MAKE IT ASCII
MOV #R0,-(R4) ;PUT IT IN THE BUFFER
RESREG
  
```



# I01

```

10368 052560 011646          MOV      (SP),-(SP)      ;SAVE RETURN PC
10369 052562 016666 000004 000002  MOV      4(SP),2(SP)    ;AND RETURN PSW
10370 052570 012766 001342 000004  MOV      $FLBUFF,4(SP) ;PUT BUFFER ADDRESS ON STACK
10371 052576 000006          RTT                    ;RETURN
10372                                     .EVEN
10373                                     ;*****
10374 .SBTTL  CONVERT FLOATING DOUBLE BINARY TO OCTAL ASCIZ
10375                                     ;*
10376                                     ;*THIS ROUTINE CONVERTS A 64 BIT FLOATING NUMBER TO AN OCTAL
10377                                     ;*ASCIZ STRING IN THE FOLLOWING FORMAT:
10378                                     ;*
10379                                     ;*      U VVV WWW XXXXXX YYYYYY ZZZZZZ
10380                                     ;*
10381                                     ;*      WHERE U = SIGN BIT
10382                                     ;*            V = 8-BIT EXPONENT (RIGHT JUSTIFIED)
10383                                     ;*            W = FRACTION BITS<57:51> (RIGHT JUSTIFIED)
10384                                     ;*            X = FRACTION BITS <50:35>
10385                                     ;*            Y = FRACTION BITS <34:19>
10386                                     ;*            Z = FRACTION BITS <18:03>
10387                                     ;*
10388                                     ;*IT IS ENTERED BY A TRAP CALL WITH THE ADDRESS OF THE FLOATING
10389                                     ;*NUMBER IN THE WORD FOLLOWING THE CALL.
10390                                     ;*IT RETURNS WITH THE ADDRESS OF THE ASCIZ STRING ON THE STACK.
10391                                     ;*****
10392 052600 104406          $FLD20: SAVREG
10393 052602 017667 000000 000006  MOV      2(SP),1$      ;GET ADDRESS OF DATA TO CONVERT
10394 052610 062716 000002          ADD      #2,(SP)      ;ADJUST RETURN PC
10395 052614 104410          FL20          ;CONVERT AS 32 BITS
10396 052616 000000          1$:      .WORD
10397 052620 012600          MOV      (SP)+,R0     ;GET ADDRESS OF CONVERTED DATA
10398 052622 010067 126560          MOV      R0,$BUFF     ;SAVE IT
10399 052626 062700 000041          ADD      #41,R0      ;ADJUST TO END OF BUFFER
10400 052632 105040          CLRB     -(R0)      ;PUT TERMINATOR IN BUFFER
10401 052634 016701 177756          MOV      1$,R1       ;GET ADDRESS OF DATA TO CONVERT
10402 052640 062701 000004          ADD      #4,R1       ;ADJUST TO LOWER 32 BITS
10403 052644 012102          MOV      (R1)+,R2    ;SAVE THE DATA
10404 052646 012103          MOV      (R1)+,R3
10405 052650 012701 000002          MOV      #2,R1       ;SET LOOP COUNT
10406 052654 012704 000005          3$:      MOV      #5,R4       ;SET LOOP COUNT
10407 052660 010305          4$:      MOV      R3,R5      ;GET LS 32 BITS OF DATA
10408 052662 042705 177770          BIC      #1<7,R5     ;MASK 3 BITS
10409 052666 062705 000060          ADD      #60,R5     ;MAKE THEM ASCII
10410 052672 110540          MOV      R5,-(R0)    ;PUT IN BUFFER
10411 052674 073227 177775          ASHC    #-3,R2      ;GET NEXT 3 BITS
10412 052700 077411          SOB     R4,4$       ;CONTINUE
10413 052702 010305          MOV      R3,R5      ;GET LS 32 BITS
10414 052704 042705 177776          BIC      #1<1,R5    ;ONLY WANT 1 BIT
10415 052710 062705 000060          ADD      #60,R5     ;MAKE IT ASCII
10416 052714 110540          MOV      R5,-(R0)    ;PUT IN TABLE
10417 052716 112740 000040          MOV      #40,-(R0)  ;PUT SPACE IN TABLE
10418 052722 073227 177777          ASHC    #-1,R2
10419 052726 077126          SOB     R1,3$       ;CONVERT NEXT 16 BITS
10420 052730 104407          RESREG
10421 052732 011646          MOV      (SP),-(SP)  ;ADJUST STACK
10422 052734 016666 000004 000002  MOV      4(SP),2(SP)  ;TO RETURN WITH ADDRESS
10423 052742 016766 126440 000004  MOV      $BUFF,4(SP) ;OF BUFFER ON STACK
  
```

10424 052750 000006  
 10425  
 10426  
 10427  
 10428  
 10429  
 10430  
 10431  
 10432  
 10433  
 10434  
 10435  
 10436  
 10437  
 10438 052752  
 10439 052752 010046  
 10440 052754 010146  
 10441 052756 010246  
 10442 052760 016700 126600  
 10443 052764 016701 126576  
 10444 052770 012702 177771  
 10445 052774 006300  
 10446 052776 006101  
 10447 053000 005202  
 10448 053002 001374  
 10449 053004 066700 126554  
 10450 053010 005501  
 10451 053012 066701 126550  
 10452 053016 062700 001057  
 10453 053022 005501  
 10454 053024 062701 047401  
 10455 053030 010067 126530  
 10456 053034 010167 126526  
 10457 053040 012602  
 10458 053042 012601  
 10459 053044 012600  
 10460 053046 000207  
 10461  
 10462  
 10463  
 10464  
 10465  
 10466  
 10467  
 10468  
 10469  
 10470 053050 012767 000002 000130  
 10471 053056 016700 000124  
 10472 053062 012702 001424  
 10473 053066 012701 000002  
 10474 053072 004767 177654  
 10475 053076 022701 000002  
 10476 053102 001404  
 10477 053104 022767 000002 000074  
 10478 053112 001407  
 10479 053114 016703 126446

```

RTT ;RETURN
;*****
.SBTTL RANDOM NUMBER GENERATOR ROUTINE
;THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
;WITH A RANGE OF 0 TO 2(+33)-1.
;CALL:
;* JSR PC,SRAND ;CALL THE ROUTINE
;* RETURN ;RETURN HERE THE RANDOM
;* ;NUMBER WILL BE IN
;* ;SHINUM,SLONUM

SRAND:
MOV RO,-(SP) ;PUSH RO ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
MOV R2,-(SP) ;PUSH R2 ON STACK
MOV SLONUM,RO ;SET RO WITH LOW
MOV SHINUM,R1 ;SET R1 WITH HIGH
MOV #7,R2 ;SET SHIFT COUNT
1$: ASL RO ;SHIFT RO LEFT AND
ROL R1 ;ROTATE CARRY INTO R1 AND
INC R2 ;CHECK FOR DONE
BNE 1$ ;CONTINUE SHIFT LOOP
ADD SLONUM,RO ;ADD NUMBER TO MAKE X 129
ADC R1 ;PROPOGATE CARRY
ADD SHINUM,R1 ;ADD NUMBER TO MAKE X 129
ADD #1057,RO ;ADD LOW CONSTANT
ADC R1 ;PROPOGATE CARRY
ADD #47401,R1 ;ADD HIGH CONSTANT
MOV RO,SLONUM ;SAVE RO
MOV R1,SHINUM ;SAVE R1
MOV (SP)+,R2 ;POP STACK INTO R2
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,RO ;POP STACK INTO RO
RTS PC ;RETURN
;*****
.SBTTL FLOATING POINT NUMBER GENERATOR
;THIS ROUTINE GENERATES TWO RANDOM FLOATING POINT NUMBERS
;IN EITHER SINGLE OR DOUBLE PRECISION. FOR SINGLE PRECISION
;THE NUMBERS ARE STORED IN FTMP0 AND FTMP2. DOUBLE PRECISION
;NUMBERS ARE STORED IN FTMP0 AND FTMP4.
;IN EITHER SINGLE OR DOUBLE THE EXTENDED EXPONENT IS STORED
;IN FREG0 AND FREG1.
;*****
FLTDBL: MOV #2,SOBDBL ;SET LOOP FOR 2, FOUR WORD NUMBERS
FLTSG: MOV SOBDBL,RO ;SET WORD LENGTH LOOP
MOV #FTMP0,R2 ;GET ADDRESS TO STORE WORDS IN
2$: MOV #2,R1 ;SET NUMBER OF WORDS TO 2
1$: JSR PC,SRAND ;GET RANDOM NUMBER
CMP #2,R1 ;FIRST TIME?
BEQ 3$ ;BRANCH IF YES
CMP #2,SOBDBL ;DOUBLE PRECISION?
BEQ 4$ ;BRANCH IF YES
3$: MOV SHINUM,R3 ;GET EXPONENT PART
    
```



K01

```

10480 053120 042703 000177      BIC      #177,R3      ;CHECK FOR MINUS ZERO
10481 053124 022703 100000      CMP      #BIT15,R3
10482 053130 001760              BEQ      15          ;BRANCH IF MINUS ZERO
10483 053132 016722 126430      45:     MOV      $HINUM,(R2)+ ;SAVE HINUM
10484 053136 016722 126422      MOV      $LONUM,(R2)+ ;SAVE LONUM
10485 053142 077125              SOB      R1,15       ;CONTINUE
10486 053144 077030              SOB      R0,25       ;CONTINUE FOR DOUBLE PREC
10487 053146 012746 001424      MOV      #FTMPO,-(SP) ;PUT ADDRESS OF NUMBER ON STACK
10488 053152 012746 001002      MOV      #1002,-(SP) ;PUT CONTROL WORD ON STACK
10489 053156 022767 000002 000022  CMP      #2,S0BDBL   ;DOUBLE PREC?
10490 053164 001002              BNE      55         ;BRANCH IF NO
10491 053166 012716 001004      MOV      #1004,(SP)  ;CHANGE CONTROL WORD
10492 053172 004767 000012      55:     JSR      PC,EXPEXT ;CALCULATE EXT EXPONENTS
10493 053176 012767 000001 000002  MOV      #1,S0BDBL   ;INIT S0BDBL FOR SINGLE PREC
10494 053204 000207              RTS      PC          ;RETURN
10495 053206 000001      S0BDBL: .WORD      1

```

```

*****
.SBTTL  FLOATING POINT EXPONENT EXTENSION
*      THIS ROUTINE CONVERTS THE ACTUAL EXPONENT OF A FLOATING POINT
*      NUMBER INTO AN ACTUAL EXPONENT OF 200 AND AN EXTENDED
*      EXPONENT EQUAL TO THE DIFFERENCE BETWEEN THE ORIGINAL
*      ACTUAL EXPONENT AND 200.
*
*      THE ROUTINE IS ENTERED WITH A CONTROL WORD ON THE STACK.
*      BIT 15 OF THE CONTROL WORD INDICATES WHETHER THE NUMBER
*      IS IN MEMORY (<15>=0) OR IN AN ACCUMULATOR (<15>=1).
*      IF THE NUMBER IS IN AN ACCUMULATOR, BITS <9:8> INDICATE
*      THE ACCUMULATOR NUMBER. IF THE NUMBER(S) IS IN MEMORY
*      BITS <9:8> INDICATE THE NUMBER OF NUMBERS TO CONVERT AND
*      BITS <2:0> INDICATE THE WORD LENGTH OF THE NUMBER(S).
*      IN THE CASE OF A MEMORY CONVERSION, THE ADDRESS OF THE
*      FIRST WORD TO CONVERT IS ALSO ON THE STACK (PRECEDING
*      THE CONTROL WORD).
*****

```

```

10514 053210 012605      EXPEXT: MOV      (SP)+,R5      ;SAVE RETURN PC
10515 053212 012600      MOV      (SP)+,R0      ;GET CONTROL WORD
10516 053214 100437      BMI      15          ;BRANCH IF ACC CONVERSION
10517 053216 012601      MOV      (SP)+,R1      ;GET START ADDRESS
10518 053220 162700 000400      SUB      #400,R0
10519 053224 012702 001424      MOV      #FTMPO,R2      ;GET OFFSET FROM FTMPO
10520 053230 160102      SUB      R1,R2
10521 053232 005402      NEG      R2
10522 053234 006202      ASR      R2
10523 053236 062702 001444      35:     ADD      #FREG0,R2      ;GEN ADDRESS OF EXT WORD
10524 053242 011103      MOV      (R1),R3      ;GET DATA
10525 053244 042703 100177      BIC      #100177,R3    ;GET EXPONENT
10526 053250 072327 177771      ASH      #-7,R3       ;RIGHT JUSTIFY EXPONENT
10527 053254 162703 000200      SUB      #200,R3      ;CONVERT TO 2'S COMPLIMENT
10528 053260 010312      MOV      R3,(R2)      ;ADD TO EXTENDED EXPONENT
10529 053262 042711 077600      BIC      #77600,(R1)   ;MAKE ACTUAL
10530 053266 052711 040000      BIS      #BIT14,(R1)  ;EXPONENT 200
10531 053272 162700 000400      SUB      #400,R0      ;ANY MORE WORDS?
10532 053276 100435      BMI      25          ;BRANCH IF NO
10533 053300 110003      MOVB     R0,R3        ;GET WORD LENGTH
10534 053302 006303      ASL      R3
10535 053304 060301      ADD      R3,R1        ;SELECT NEXT NUMBER ADDRESS

```

LO1

```

10536 053306 062702 000002          ADD      #2,R2          ;SELECT NEXT EXTENDED ADDRESS
10537 053312 000753          BR       3$           ;CONTINUE
10538                                     ;ACCUMULATOR CONVERSION
10539 053314 072027 177776 1$:    ASH      #-2,R0          ;GET ACCUMULATOR NUMBER
10540 053320 042700 177477          BIC      #177477,R0
10541 053324 010002          MOV      R0,R2
10542 053326 072227 177773          ASH      #-5,R2          ;GENERATE ADDRESS OF
10543 053332 062702 001410          ADD      #SAC0,R2        ;EXTENDED EXPONENT
10544 053336 042767 000300 000004  BIC      #300,5$        ;GENERATE INSTRUCTION
10545 053344 050067 000000          BIS      R0,5$         ;TO GET EXPONENT
10546 053350 175003          STEXP   AC0,R3        ;GET EXPONENT
10547 053352 060312          ADD      R3,(R2)       ;ADD TO EXTENDED EXPONENT
10548 053354 005003          CLR      R3
10549 053356 042767 000300 000004  BIC      #300,4$        ;GENERATE INSTRUCTION
10550 053364 050067 000000          BIS      R0,4$         ;TO LOAD EXPONENT BACK TO ACC
10551 053370 176403          LDEXP   R3,AC0        ;LOAD EXPONENT OF 200
10552 053372 010546 2$:    MOV      R5,-(SP)      ;RESTORE RETURN PC
10553 053374 000207          RTS      PC           ;RETURN
10554                                     ;.SBTTL POWER DOWN AND UP ROUTINES
10555
10556                                     ;*****
10557                                     ;POWER DOWN ROUTINE
10558 053376 012737 053550 000024 $PWRDN: MOV      #SILLUP,2#PWRVEC ;SET FOR FAST UP
10559 053404 012737 000340 000026  MOV      #340,2#PWRVEC+2 ;PRIO:7
10560 053412 010046          MOV      R0,-(SP)      ;PUSH R0 ON STACK
10561 053414 010146          MOV      R1,-(SP)      ;PUSH R1 ON STACK
10562 053416 010246          MOV      R2,-(SP)      ;PUSH R2 ON STACK
10563 053420 010346          MOV      R3,-(SP)      ;PUSH R3 ON STACK
10564 053422 010446          MOV      R4,-(SP)      ;PUSH R4 ON STACK
10565 053424 010546          MOV      R5,-(SP)      ;PUSH R5 ON STACK
10566 053426 017746 125610  MOV      2$SWR,-(SP)    ;PUSH 2$SWR ON STACK
10567 053432 010667 000116  MOV      SP,$SAVR6     ;SAVE SP
10568 053436 012737 053450 000024  MOV      #2$PWRUP,2#PWRVEC ;SET UP VECTOR
10569 053444 000000          HALT
10570 053446 000776          BR       #-2          ;;HANG UP
10571
10572                                     ;*****
10573                                     ;POWER UP ROUTINE
10574 053450 012737 053550 000024 $PWRUP: MOV      #SILLUP,2#PWRVEC ;SET FOR FAST DOWN
10575 053456 016706 000072          MOV      $SAVR6,SP    ;GET SP
10576 053462 005067 000066          CLR      $SAVR6       ;WAIT LOOP FOR THE TTY
10577 053466 005267 000062 1$:    INC      $SAVR6       ;WAIT FOR THE INC
10578 053472 001375          BNE     1$            ;OF WORD
10579 053474 011600          MOV      (SP),R0      ;GET THE OLD SW. REG. VALUE
10580 053476 076600          MED     ;WRITE THE CONSOLE SW. REG.
10581 053500 000226          WCNSSW ;WITH ITS PREVIOUS VALUE
10582 053502 012677 125534          MOV      (SP)+,2$SWR  ;POP STACK INTO 2$SWR
10583 053506 012605          MOV      (SP)+,R5     ;POP STACK INTO R5
10584 053510 012604          MOV      (SP)+,R4     ;POP STACK INTO R4
10585 053512 012603          MOV      (SP)+,R3     ;POP STACK INTO R3
10586 053514 012602          MOV      (SP)+,R2     ;POP STACK INTO R2
10587 053516 012601          MOV      (SP)+,R1     ;POP STACK INTO R1
10588 053520 012600          MOV      (SP)+,R0     ;POP STACK INTO R0
10589 053522 012737 053376 000024  MOV      #2$PWRDN,2#PWRVEC ;SET UP THE POWER DOWN VECTOR
10590 053530 012737 000340 000026  MOV      #340,2#PWRVEC+2 ;PRIO:7
10591 053536 104401          TYPE ;REPORT THE POWER FAILURE

```



MO1

```
10592 053540 053556 SPWRMG: .WORD SPOWER ;;POWER FAIL MESSAGE POINTER
10593 053542 012716 MOV (PC)+,(SP) ;;RESTART AT START
10594 053544 003276 SPWRAD: .WORD START ;;RESTART ADDRESS
10595 053546 000002 RTI
10596 053550 000000 SILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
10597 053552 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
10598 053554 000000 S$AVR6: 0 ;;PUT THE SP HERE
10599 053556 005015 SPOWER: .ASCIZ <15><12>"POWER"
10600 053564 000122
```

047520 042527

```
10601 .EVEN
10602 .SBTTL TRAP DECODER
10603
10604 ;*****
10605 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
10606 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
10607 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
10608 ;*GO TO THAT ROUTINE.
```

```
10609
10610 053566 010046 STRAP: MOV RO,-(SP) ;;SAVE RO
10611 053570 016600 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS
10612 053574 005740 TST -(RO) ;;BACKUP BY 2
10613 053576 111000 MOV (RO),RO ;;GET RIGHT BYTE OF TRAP
10614 053600 006300 ASL RO ;;POSITION FOR INDEXING
10615 053602 016000 053622 MOV $TRPAD(RO),RO ;;INDEX TO TABLE
10616 053606 000200 RTS RO ;;GO TO ROUTINE
```

```
10617
10618
10619 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
10620
10621 053610 011646 STRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
10622 053612 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
10623 053620 000002 RTI ;;RESTORE THE PSW
```

```
10624
10625 .SBTTL TRAP TABLE
10626
10627 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
10628 ;*BY THE "TRAP" INSTRUCTION.
```

```
10629
10630 ; ROUTINE
10631 ;-----
10632 053622 053610 STRPAD: .WORD STRAP2
10633 053624 050562 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
10634 053626 051432 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
10635 053630 051406 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
10636 053632 051446 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
10637 053634 051634 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
```

```
10638
10639
10640 053636 052216 $SAVREG ;;CALL=SAVREG TRAP+6(104406) SAVE RO-R5 ROUTINE
10641 053640 052254 $RESREG ;;CALL=RESREG TRAP+7(104407) RESTORE RO-R5 ROUTINE
10642 053642 052312 $FL20 ;;CALL=FL20 TRAP+10(104410)
10643 053644 052600 $FLD20 ;;CALL=FLD20 TRAP+11(104411)
```

```
10644 ;*****
10645 .SBTTL UNIBUS EXERCISER INITIALIZATION ROUTINE
10646 ;*THIS ROUTINE INITIALIZES THE BASE ADDRESS FOR THE
10647 ;*UNIBUS EXERCISER AND LOADS UP THE EXERCISER REGISTERS.
```

```

10648
10649 053646 012703 001716
10650 053652 005023
10651 053654 005023
10652 053656 005023
10653 053660 005013
10654
10655
10656
10657 053662 012702 002304
10658 053666 005072 000010
10659 053672 012772 045730 000012
10660 053700 012772 000340 000014
10661 053706 012732 170000
10662
10663 053712 013732 001722
10664 053716 013732 001724
10665 053722 000207

;*****
UBEINIT:MOV #UBESAV,R3 ;GET ADDRESS OF NEXT UBE ADRES
          CLR (R3)+ ;INITIALIZE
          CLR (R3)+
          CLR (R3)+
          CLR (R3)

;SET UP THE UBE AND START IT
MOV #UBETBL,R2 ;GET ADDRESS OF UBE TABLE
CLR @10(R2) ;CLEAR ALL ERRORS
MOV #UBESRV,@12(R2) ;SET UP UBE VECTOR
MOV #PR7,@14(R2) ;SET UP UBE VECTOR PSW
MOV #170000,@(R2)+ ;SET CC FOR 2K WORD TRANSFER
;UBE IS DOING BYTE TRANSFERS
MOV @#UBEADR,@(R2)+ ;LOAD UBE BUS ADDRESS
MOV @#UBEADR+2,@(R2)+ ;LOAD ADR BITS 16 & 17
RTS PC ;RETURN

```



```

10666
10667
10668
10669
10670
10671
10672
10673
10674
10675
10676
10677
10678
10679
10680
10681
10682
10683
10684 053724 104406
10685 053726 013703 001536
10686 053732 105737 001545
10687 053736 001426
10688 053740 005002
10689 053742 073227 000003
10690 053746 072327 177775
10691 053752 042703 160000
10692 053756 006102
10693 053760 062702 172340
10694 053764 011205
10695 053766 005004
10696 053770 073427 000006
10697 053774 060305
10698 053776 005504
10699 054000 010437 001542
10700 054004 010537 001540
10701 054010 104407
10702 054012 000207
10703 054014 163703 001550
10704 054020 005004
10705 054022 010305
10706 054024 000765
10707
10708
10709
10710
10711
10712
10713
10714
10715
10716
10717
10718
10719
10720 054026 012737 054070 000114
10721 054034 024042

```

```

*****
:SBTTL CONVERT A VIRTUAL ADDRESS TO A PHYSICAL ADDRESS
: THIS ROUTINE CONVERTS A 16-BIT VIRTUAL ADDRESS TO A
: 18-BIT PHYSICAL ADDRESS. THE VIRTUAL ADDRESS IS
: ASSUMED TO BE IN LOCATION "VADR" AND THE PHYSICAL
: ADDRESS IS PLACED IN LOCATIONS "PA1716" AND "PA1500".
:
: IF MEMORY MANAGEMENT IS OFF THE PHYSICAL ADDRESS IS
: GENERATED BY SUBTRACTING THE CONTENTS OF LOCATION
: "SFACOR" FROM THE VIRTUAL ADDRESS. THIS LOCATION
: CONTAINS THE BYTE OFFSET BETWEEN THE RELOCATED CODE
: AND THE NON-RELOCATED CODE.
:
: IF MEMORY MANAGEMENT IS ON, THE CONTENTS OF THE
: APPROPRIATE PAR REGISTER IS ADDED(AFTER ADJUSTMENT)
: TO THE LEAST SIGNIFICANT 13 BITS OF THE VIRTUAL ADDRESS.
*****

```

```

CNVADR: SAVREG
MOV      2#VADR,R3      ;GET VIRTUAL ADDRESS TO CONVERT
TSTB    2#MMON         ;IS MEMORY MGMT ON?
BEQ     1$             ;BRANCH IF NO
CLR     R2
ASHC    #3,R2          ;GET PAR SELECT BITS
ASH     #-3,R3         ;RETURN VIR ADDR TO ORIGINAL
BIC     #160000,R3    ;MAKE SURE SIGN DIDN'T EXTEND
ROL     R2             ;MAKE R2 EVEN FOR WORD ADDRESSING
ADD     #KIPARO,R2    ;GET ADDRESS OF PAR
MOV     (R2),R5       ;GET PAR DATA
CLR     R4             ;SETUP R4
ASHC    #6,R4         ;SHIFT PAR DATA
ADD     R3,R5         ;FORM PHYSICAL ADDRESS
2$: MOV   R4,2#PA1716  ;SAVE PHYSICAL
MOV     R5,2#PA1500  ;ADDRESS
RESREG
RTS     PC            ;RETURN
1$: SUB  2#FACTOR,R3  ;FORM PHYSICAL ADDRESS
CLR     R4
MOV     R3,R5
BR     2$            ;RETURN

```

```

*****
:SBTTL ROUTINE TO CHECK RELOCATED DATA
:ROUTINE TO CHECK DATA RELOCATED
:CALL: R0= HIGHEST ADDRESS +2 OF SOURCE DATA
:      R2= HIGHEST ADDRESS +2 OF DEST DATA
:      R5= LOWEST ADDRESS OF THE SOURCE DATA
:
: THIS ROUTINE USES A COMPARE INSTRUCTION TO CHECK
: THE DATA THAT WAS RELOCATED. IF A PARITY ERROR OCCURS
: DURING THIS CHECK A SPECIAL ERROR MESSAGE IS TYPED
: INSTEAD OF THE UNEXPECTED TRAP MESSAGE.
*****
CHKDAT: MOV  #2$,2#CACHVEC ;SETUP PARITY VECTOR.
CMP      -(R0),-(R2)      ;CHECK DATA

```

```

10722 054036 001012          BNE      99$
10723 054040 005112          COM      (R2)          ;COMPLEMENT DEST DATA
10724 054042 005112          COM      (R2)          ;TWICE
10725 054044 021210          CMP      (R2), (R0)   ;CHECK DATA
10726 054046 001006          BNE      99$
10727 054050 020005          1$:    CMP      R0, R5          ;BRANCH IF ALL DATA NOT CHECKED
10728 054052 001365          BNE      CHKDAT
10729 054054 012737 054414 000114  MOV      #.PARSRV, @#CACHVEC ;RESTORE CACHVEC
10730 054062 000207          RTS      PC          ;RETURN
10731 054064 000262          99$:   SEV
10732 054066 000207          RTS      PC
10733 054070 013737 177744 001740  2$:   MOV      @#MEMERR, @#REPREG ;SAVE ERROR REG
10734 054076 010237 001536          MOV      R2, @#VADR
10735 054102 010037 001750          MOV      R0, @#SRCADR
10736 054106 104005          ERROR   5
10737 054110 000765          BR       99$          ;RETURN
10738
10739
10740          ;*****
10741          ;SBTTL ROUTINE TO CLEAR 'T' BIT
10742          ;*****
10743 054112 013746 177776  CLRTRBIT: MOV      @#PSW, -(SP) ;PUSH PSW ONTO STACK
10744 054116 011627          MOV      (SP), (PC)+ ;SAVE IN RETPSW BELOW
10745 054120 000000          RETPSW: .WORD   0
10746 054122 042716 000020          BIC      #20, (SP) ;CLEAR T BIT IN PSW ON STACK
10747          ;*****
10748          ;SBTTL ROUTINE TO RESTORE THE T BIT
10749          ;*****
10750 054126 012746 054134  RESPSW: MOV      #1$, -(SP) ;SET RETURN PC FOR RTI
10751 054132 000002          RTI
10752 054134 000207          1$:    RTS      PC          ;RETURN
10753
10754 054136 042737 177400 177776  RESTPS: BIC      #177400, @#PSW ;SET KERNEL MODE
10755 054144 016746 177750          MOV      RETPSW, -(SP) ;PUSH ORIG PSW ONTO STACK
10756 054150 000766          BR       RESPSW
10757
10758          ;*****
10759          ;SBTTL KEYBOARD INT SERV ROUTINE
10760          ;THIS ROUTINE HANDLES INTERRUPTS FROM THE KEYBOARD
10761          ;
10762          ;TYPING A CONTROL 'C' WILL CAUSE THE PROCESSOR TO HALT
10763          ;
10764          ;TYPING A CARRAGE RETURN WILL CAUSE A CARRIAGE RETURN-LINE FEED
10765          ;TO BE TYPED.
10766          ;
10767          ;TYPING A CONTROL 'O' WILL INHIBIT ANY FURTHER TYPEOUT. THE SECOND CONTROL 'O'
10768          ;WILL ENABLE TYPEOUT AGAIN AND ECHO A CR-LF.
10769          ;
10770          ;ANY OTHER CHARACTER WILL JUST BE ECHOED.
10771          ;*****
10772
10773          000003          CNTRLC=3
10774          000017          CNTRLO=17
10775
10776 054152 017746 125072  TKISR:  MOV      @#STKB, -(SP) ;GET CHARACTER
10777 054156 042716 177600          BIC      #177600, (SP) ;STRIP UNUSED BITS

```



```

10778 054162 022716 000003      CMP      #CNTRLC,(SP)      ;BRANCH IF NOT CONTROL C (1C)
10779 054166 001010      BNE      1$              ;
10780 054170 012737 001334 001276  MOV      #SCRLF-1,@#SREG5 ;ECHO CR LF
10781 054176 106277 125050      ASRB     @STPS
10782 054202 005726      TST      (SP)+          ;POP CHARACTER OFF THE STACK
10783 054204 000000      HALT
10784 054206 000002      RTI                    ;RETURN
10785
10786 054210 122716 000015      1$:      CMPB     #15,(SP)      ;BRANCH IF NOT <CR>
10787 054214 001007      BNE      2$              ;
10788 054216 012737 001334 001276  MOV      #SCRLF-1,@#SREG5 ;ECHO CR LF
10789 054224 106277 125022      ASRB     @STPS
10790 054230 005726      TST      (SP)+          ;POP CHARACTER OFF STACK
10791 054232 000002      RTI                    ;RETURN
10792
10793 054234 122716 000017      2$:      CMPB     #CNTRLO,(SP)   ;BRANCH IF NOT CONTROL 0 (10)
10794 054240 001012      BNE      3$              ;
10795 054242 005726      TST      (SP)+          ;
10796 054244 005167 125260      COM      NOTYPE
10797 054250 100405      BMI      7$              ;
10798 054252 012737 001334 001276  MOV      #SCRLF-1,@#SREG5 ;ECHO CR LF
10799 054260 106277 124766      ASRB     @STPS
10800 054264 000002      RTI                    ;
10801
10802 054266 104406      3$:      SAVREG
10803 054270 011605      MOV      (SP),R5        ;RETRIEVE CHARACTER
10804 054272 004737 054516      JSR      PC,@#LDKT      ;GO TO LOW CORE
10805 054276 013700 001504      MOV      @#TKBFRP,R0    ;GET BUFFER PTR
10806 054302 110520      4$:      MOV      R5,(R0)+      ;LOAD CHAR INTO BFR
10807 054304 105010      CLRB     (R0)          ;CLEAR NEXT LOC
10808 054306 022700 001526      5$:      CMP      #TKBFR+20,R0   ;BRANCH IF NOT END OF BFR
10809 054312 001002      BNE      6$              ;
10810 054314 012700 001506      MOV      #TKBFR,R0      ;RESET BUFFER PTR
10811 054320 010037 001504      6$:      MOV      R0,@#TKBFRP    ;RESTORE BFR PTR
10812 054324 004737 054604      JSR      PC,@#RESKT     ;GO BACK TO ORIGINAL MEMORY
10813 054330 104407      RESREG
10814 054332 005737 001530      ECHO:    TST      @#NOTYPE     ;TYPEOUT DISABLED?
10815 054336 100004      BPL      1$              ;BRANCH IF NO
10816 054340 005726      TST      (SP)+          ;FIX UP STACK
10817 054342 105077 124704      CLRB     @STPS         ;CLEAR IE BIT
10818 054346 000002      RTI                    ;RETURN
10819 054350 105777 124676      1$:      TSTB     @STPS         ;PRINTER READY?
10820 054354 100375      BPL      -4              ;BRANCH IF NO
10821 054356 112677 124672      MOV      (SP)+,@STPB    ;MOVE CHAR TO PRINTER
10822 054362 000002      RTI                    ;RETURN
10823
10824
10825
10826
10827
10828
10829
10830 054364 005237 001276      *****
10831 054370 117746 124702      ;SBTTL TELETYPE INTERRUPT SERVICE ROUTINE
10832 054374 001356      ;*THIS ROUTINE TYPES A MESSAGE POINTED TO BY THE ADR STORED
10833 054376 005726      ;*IN LOCATION SREG5. THIS ROUTINE IS INTERRUPT DRIVEN.
10834
10835
10836
10837
10838
10839
10840
10841
10842
10843
10844
10845
10846
10847
10848
10849
10850
10851
10852
10853
10854
10855
10856
10857
10858
10859
10860
10861
10862
10863
10864
10865
10866
10867
10868
10869
10870
10871
10872
10873
10874
10875
10876
10877
10878
10879
10880
10881
10882
10883
10884
10885
10886
10887
10888
10889
10890
10891
10892
10893
10894
10895
10896
10897
10898
10899
10900
10901
10902
10903
10904
10905
10906
10907
10908
10909
10910
10911
10912
10913
10914
10915
10916
10917
10918
10919
10920
10921
10922
10923
10924
10925
10926
10927
10928
10929
10930
10931
10932
10933
10934
10935
10936
10937
10938
10939
10940
10941
10942
10943
10944
10945
10946
10947
10948
10949
10950
10951
10952
10953
10954
10955
10956
10957
10958
10959
10960
10961
10962
10963
10964
10965
10966
10967
10968
10969
10970
10971
10972
10973
10974
10975
10976
10977
10978
10979
10980
10981
10982
10983
10984
10985
10986
10987
10988
10989
10990
10991
10992
10993
10994
10995
10996
10997
10998
10999
11000

```

E02

10834 054400 005077 124646  
10835 054404 012737 001570 001276  
10836 054412 000002  
10837  
10838  
10839  
10840  
10841  
10842  
10843  
10844  
10845  
10846  
10847  
10848  
10849  
10850  
10851  
10852  
10853  
10854 054414  
10855 054414 005227  
10856 054416 177777  
10857 054420 001401  
10858 054422 000000  
10859  
10860  
10861  
10862 054424  
10863 054424 016637 000002 001736  
10864 054432 011637 001536  
10865 054436 162737 000002 001536  
10866 054444 013737 177744 001740  
10867 054452 013737 001212 001742  
10868 054460 012737 054470 001212  
10869 054466 104004  
10870 054470 013737 001742 001212  
10871 054476 012767 177777 177712  
10872 054504 012716 054512  
10873  
10874 054510 000002  
10875 054512 000177 124474  
10876  
10877  
10878  
10879  
10880  
10881  
10882 054516 105737 001545  
10883 054522 001427  
10884 054524 012604  
10885 054526 013737 177776 054660  
10886 054534 042737 140000 177776  
10887 054542 012700 172340  
10888 054546 012001  
10889 054550 012002

```
CLR      2STPS      ;CLEAR IE BIT
MOV      2NULLS,2#SREG5
RTI      ;RETURN

;*****
;SBTTL  PARITY ERROR SERVICE
;      THIS ROUTINE FIELDS UNEXPECTED TRAPS TO 114. IT IS ASSUMED
;      THAT THE ERROR WAS IN CACHE AND WAS CAUSED BY THE "OTHER
;      WORD" RATHER THAN THE "WANTED WORD" WHICH MEANS THAT THE
;      BAD DATA IS STILL IN THE CACHE. SO, TO CLEAR THE BAD DATA
;      THE ERROR ADDRESS IS REFERENCED CAUSING THE CACHE TO GO
;      TO MAIN MEMORY TO GET THE DATA. THIS PREVENTS AN
;      ARBITRARY REFERENCE TO THE BAD WORD FROM TRAPPING.
;
;      AFTER THE ERROR IS REPORTED, BITS 2 AND 3 OF THE MEMORY
;      ERROR REGISTER ARE TESTED TO SEE IF THE BAD DATA IS IN
;      MAIN MEMORY. IF IT IS, THE PROGRAM RESTARTS SINCE THE
;      GOOD DATA IS NOW LOST FOREVER. OTHERWISE THE PROGRAM
;      RETURNS TO THE ADDRESS POINTED TO BY "SLPERR".
;*****
.PARSRV:
PARFLG:  INC      (PC)+      ;MAKE FLAG ZERO FIRST TIME THRU
        .WORD    -1         ;NEGATIVE ONE FOR A FLAG
        BEQ     55         ;BRANCH IF FIRST TIME IN
        HALT    ;I HAVE ENTERED THIS ROUTINE BEFORE
                ;I FINISHED REPORTING THE FIRST ERROR
                ;THE SECOND ENTRY ADDRESS IS ON THE
                ;STACK
55:     MOV      2(SP),2#REPPSW ;SAVE ERROR PSW
        MOV      (SP),2#VADR   ;SAVE PC
        SUB     2,2#VADR      ;ADJUST ERROR PC
PERET:  MOV      2#MEMERR,2#REPREG ;SAVE ERROR REG FOR TYPEOUT
        MOV      2#SLPERR,2#REPADR ;SAVE LOOP ADDRESS
        MOV      25,2#SLPERR   ;SET RETURN ADDRESS IF LOOPING
        ERROR   4
25:     MOV      2#REPADR,2#SLPERR ;RESTORE LOOP ADDRESS
15:     MOV      #-1,PARFLG    ;RESTORE NEGATIVE ONE FOR A FLAG
        MOV      #X,(SP)      ;PUT ADDRESS ON STACK TO GET ORIGINAL
                ;PSW BACK
RT1:    RTI      ;GET OLD PSW
X:      JMP     2#SLPERR      ;JUMP TO START OF TEST THAT HAD THE PE
;*****
;SBTTL  CONTEXT SWITCH DOWN SUBROUTINE
;      SUBROUTINE TO SAVE & LOAD KIPAR'S 0,1, AND 2 (IF MEM MGMT ENABLED)
;      THIS ROUTINE IS CALLED BY THE KEYBOARD INTERRUPT, LINE CLOCK
;      INTERRUPT, LBE SERVICE ROUTINE, MBT SERVICE ROUTINE, AND TYPE TIME ROUTINE.
;*****
L0KT:   TSTB    2#MMON        ;BRANCH IF MEM MGMT DISABLED
        BEQ     15
        MOV     (SP)+,R4      ;SAVE RETURN PC
        MOV     2#PSW,2#SSAVPSW ;SAVE THE CURRENT PSW
        BIC    2#140000,2#PSW ;GO TO KERNEL MODE
        MOV     #KIPAR,R0     ;GET ADDRESS OF PAR0
        MOV     (R0)+,R1      ;GET PAR0
        MOV     (R0)+,R2      ;GET PAR1
```



```

10890 054552 012003          MOV      (RO)+,R3          ;GET PAR2
10891 054554 012740 000400  MOV      #400,-(RO)       ;RELOC BACK TO LOW CORE
10892 054560 012740 000200  MOV      #200,-(RO)
10893 054564 005040          CLR      -(RO)
10894 054566 012700 054652  MOV      #SSAVPAR,RO     ;GET ADDRESS OF SAVE BUFFER
10895 054572 010120          MOV      R1,(RO)+        ;PUT PAR DATA IN MEMORY
10896 054574 010220          MOV      R2,(RO)+
10897 054576 010320          MOV      R3,(RO)+
10898 054600 010446          MOV      R4,-(SP)       ;PUT RETURN PC ON STACK
10899 054602 000207          1$:      RTS      PC
10900
10901          ;:*****
10902          .SBTTL CONTEXT SWITCH UP SUBROUTINE
10903          ;SUBROUTINE TO RESTORE KIPAR0, 1, AND 2 (IF MGMT ENABLED)
10904          ;:*****
10905 054604 105737 001545  RESKT:  TSTB   @#MMON          ;BRANCH IF MEM MGMT DISABLED
10906 054610 001417          BEQ      1$
10907 054612 012604          MOV      (SP)+,R4        ;GET RETURN PC
10908 054614 012700 054652  MOV      #SSAVPAR,RO     ;GET ADDRESS OF SAVE BUFF
10909 054620 012001          MOV      (RO)+,R1        ;GET OLD PAR DATA
10910 054622 012002          MOV      (RO)+,R2
10911 054624 012003          MOV      (RO)+,R3
10912 054626 012700 172340  MOV      @KIPAR0,RO      ;GET ADDRESS OF PAR0
10913 054632 010120          MOV      R1,(RO)+        ;RELOCATE BACK
10914 054634 010220          MOV      R2,(RO)+
10915 054636 010310          MOV      R3,(RO)
10916 054640 013737 054660 177776  MOV      @#SSAVPSW,@#PSW
10917 054646 010446          MOV      R4,-(SP)
10918 054650 000207          1$:      RTS      PC
10919 054652 000003          $SAVPAR:.BLKW 3
10920 054660 000000          $SAVPSW:.WORD
10921          ;:*****
10922          .SBTTL KT ABORT SUBROUTINE
10923          ;:*****
10924 054662 005227          KTABRT: INC   (PC)+        ;MAKE FLAG ZERO FIRST TIME THRU
10925 054664 177777          KTFLG:  .WORD -1          ;NEGATIVE ONE FOR A FLAG
10926 054666 001401          BEQ      5$            ;BRANCH IF FIRST TIME IN
10927 054670 000000          HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                            ;I FINISHED REPORTING THE FIRST ERROR
                            ;THE SECOND ENTRY ADDRESS IS ON THE
                            ;STACK
10928
10929
10930
10931 054672          5$:
10932 054672 016637 000002 001736  MOV      2(SP),@#REPPSW   ;SAVE ERROR PSW
10933 054700 011637 001536          MOV      (SP),@#VADR     ;SAVE ERROR PC
10934 054704 162737 000002 001536  SUB      #2,@#VADR
10935 054712 013737 177572 001744  MOV      @#MMR0,@#REPMR0  ;SAVE MMR0
10936 054720 013737 177576 001746  MOV      @#MMR2,@#REPMR2  ;SAVE MMR2
10937 054726 013737 001212 001742  MOV      @#SLPERR,@#REPADR ;SAVE LOOP ADDRESS
10938 054734 012737 054744 001212  MOV      @1$,@#SLPERR    ;SET RETURN ADR IF LOOPING
10939 054742 104003          ERROR 3
10940 054744 013737 001742 001212  1$:      MOV      @#REPADR,@#SLPERR ;RESTORE LOOP ADR
10941 054752 042737 170000 177572  BIC      #170000,@#MMR0  ;CLEAR ERRORS
10942 054760 012767 177777 177676  MOV      #-1,KTFLG      ;RESTORE NEGATIVE ONE FOR A FLAG
10943 054766 013716 001212          MOV      @#SLPERR,(SP)  ;GET LOOP ADDRESS
10944          RTI                    ;RETURN
10945

```

```

10946
10947
10948
10949 054774
10950 054774 005227
10951 054776 177777
10952 055000 001401
10953 055002 000000
10954
10955
10956
10957 055004
10958 055004 016637 000002 001736
10959 055012 011637 001536
10960 055016 162737 000002 001536
10961 055024 013737 001212 001742
10962 055032 012737 055042 001212
10963 055040 104002
10964 055042 013737 001742 001212
10965 055050 012767 177777 177720
10966 055056 013716 001212
10967 055062 000002
10968
10969
10970
10971
10972 055064 005227
10973 055066 177777
10974 055070 001401
10975 055072 000000
10976
10977
10978
10979 055074
10980 055074 016637 000002 001736
10981 055102 011637 001536
10982 055106 162737 000002 001536
10983 055114 012706 000700
10984 055120 013737 177766 001740
10985 055126 013737 001212 001740
10986 055134 012737 055144 001212
10987 055142 104001
10988 055144 013737 001740 001212
10989 055152 012767 177777 177706
10990 055160 013746 001736
10991 055164 013746 001212
10992 055170 000002
10993
10994
10995
10996
10997
10998
10999 055172 005267 000002
11000
11001 055176 000002

```

```

*****
.SBTTL RESERVED INSTRUCTION ROUTINE
*****
RESERR:
RESFLG: INC (PC)+ ;MAKE FLAG ZERO FIRST TIME THRU
        .WORD -1 ;NEGATIVE ONE FOR A FLAG
        BEQ 55 ;BRANCH IF FIRST TIME IN
        HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
              ;I FINISHED REPORTING THE FIRST ERROR
              ;THE SECOND ENTRY ADDRESS IS ON THE
              ;STACK
SS:
        MOV 2(SP),@REPPSW ;SAVE PSW
        MOV (SP),@VADR ;SAVE ERROR PC
        SUB #2,@VADR
        MOV @SLPERR,@REPADR ;SAVE LOOP ADR
        MOV #15,@SLPERR ;SET RETURN ADR IF LOOPING
        ERROR 2
IS:
        MOV @REPADR,@SLPERR ;RESTORE LOOP ADR
        MOV #-1,RESFLG ;RESTORE NEGATIVE ONE FOR A FLAG
        MOV @SLPERR,(SP) ;GET LOOP ADDRESS
        RTI ;RETURN
*****
.SBTTL TRAP TO 4 SERVICE ROUTINE
*****
ERPR: INC (PC)+ ;MAKE FLAG ZERO FIRST TIME THRU
ERFLG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
        BEQ 55 ;BRANCH IF FIRST TIME IN
        HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
              ;I FINISHED REPORTING THE FIRST ERROR
              ;THE SECOND ENTRY ADDRESS IS ON THE
              ;STACK
SS:
        MOV 2(SP),@REPPSW ;SAVE ERROR PSW
        MOV (SP),@VADR ;SAVE ERROR PC
        SUB #2,@VADR
        MOV @USESTK,SP ;RESTORE SP
        MOV @CPUERR,@REPREG ;GET ERROR REG
        MOV @SLPERR,@REPREG ;SAVE LOOP ADR
        MOV #15,@SLPERR ;SET RETURN ADR IF LOOPING
        ERROR 1
IS:
        MOV @REPREG,@SLPERR ;SET LOOP ADR
        MOV #-1,ERFLG ;RESTORE NEGATIVE ONE FOR A FLAG
        MOV @REPPSW,-(SP) ;SETUP STACK TO RETURN
        MOV @SLPERR,-(SP)
        RTI ;RETURN
*****
.SBTTL MICROBREAK TRAP SERVICE ROUTINE
*****
BKROUT: INC BKFLAG ;SET MICROBREAK FLAG TO
        RTI ;INDICATE TRAP TO 4 OCCURRED
              ;RETURN FROM TRAP

```



H02

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 227  
DQKDCR.P11 07-FEB-77 09:58 MICROBREAK TRAP SERVICE ROUTINE

11002 055200 000000  
11003

BKFLAG: .WORD 0

;MICROBREAK TRAP FLAG

```

11004
11005
11006
11007
11008
11009
11010 055202 000000
11011 055204 000020
11012 055206 140000
11013 055210 140020
11014
11015
11016 055212 002154
11017 055214 002176
11018 055216 000000
11019 055220 000000
11020 055222 002216
11021 055224 002256
11022 055226 000000
11023 055230 000000
11024 055232 002322
11025 055234 002304
11026 055236 055503
11027 055240 055511
11028 055242 056320
11029 055244 056320
11030 055246 055517
11031 055250 055525
11032 055252 056320
11033 055254 056320
11034 055256 056014
11035 055260 056357
11036 055262 046200 053517 046040
11037 055270 046511 000077
11038 055274 044510 044107 046040
11039 055302 046511 000077
11040 055306 051105 047522 050122
11041 055314 020103 044120 051531
11042 055322 050055 020103 050040
11043 055330 053523 020040 020040
11044 055336 042524 052123 047040
11045 055344 020117 052523 050102
11046 055352 051501 050055 051501
11047 055360 020123 047103 000124
11048 055366 044124 020105 047506
11049 055374 046114 053517 047111
11050 055402 020107 042504 044526
11051 055410 042503 020123 047101
11052 055416 020104 051104 053111
11053 055424 051505 053440 046111
11054 055432 020114 042502 052440
11055 055440 042523 020104 047506
11056 055446 020122 042522 047514
11057 055454 040503 044524 047117
11058 055462 100072
11059 055464 042504 044526 042503
    
```

```

;THE BELOW TABLE REPRESENTS THE 'NEW' PSW SET BY THE PROGRAM ON
;SUCCESSIVE SUB-PASSES.
;NOTE THE BELOW TABLE MAY BE MODIFIED TO CAUSE THE PROGRAM TO RUN
;UNDER USER DEFINED PARAMETERS BY PATCHING IN THE DESIRED PASS PARAMETER
;FOR EXAMPLE TO CAUSE THE PROGRAM TO RUN WITHOUT SETTING THE 'T' BIT
;IN ALL PASSES PATCH OUT THE 'T' BIT IN THE TABLE.
    
```

```

PSWTAB: 000000
        000020
        140000
        140020
;T-BIT TRAPPING
;USER MODE
;USER MODE, T-BIT TRAPPING
    
```

```

;MESSAGES
REGINX: .EVEN
        RP3DS
        RKDS
        .WORD
        .WORD
        RP4CS1
        RSCS1
        .WORD
        .WORD
        MBTTBL
        UBETBL
    
```

```

MSGINX: .WORD MSG5
        .WORD MSG6
        .WORD MSG21
        .WORD MSG21
        .WORD MSG10
        .WORD MSG11
        .WORD MSG21
        .WORD MSG21
        .WORD MSG15
        .WORD MSG24
MSG1:   .ASCIZ <CRLF>'LOW LIM?'
MSG2:   .ASCIZ 'HIGH LIM?'
MSG3:   .ASCIZ /ERRORPC PHYS-PC PSW TEST NO SUBPAS-PASS CNT/
    
```

```

MSG4:   .ASCII /THE FOLLOWING DEVICES AND DRIVES WILL BE USED FOR RELOCATION:/<CRLF>
    
```

```

.ASCIZ /DEVICE DRIVES/<CRLF>
    
```



J02

11060	055472	042011	044522	042526																
11061	055500	100123	000																	
11062	055503	122	030120	004463	MSG5:	.ASCIZ	?RPO3	?												
11063	055510	000																		
11064	055511	122	030113	004465	MSG6:	.ASCIZ	?RK05	?												
11065	055516	000																		
11066	055517	122	030120	004464	MSG10:	.ASCIZ	?RPO4	?												
11067	055524	000																		
11068	055525	122	030123	004464	MSG11:	.ASCIZ	?RS04	?												
11069	055532	000																		
11070	055533	104	053122	052123	MSG12:	.ASCIZ	/DRVSTA	ERRREG	CSREG	WRDCNT	BUSADR	DSKADR	CYLADR(RPO3)	PHYS	BUSA					
11071	055540	020101	042440	051122																
11072	055546	042522	020107	041440																
11073	055554	051123	043505	020040																
11074	055562	053440	042122	047103																
11075	055570	020124	041040	051525																
11076	055576	042101	020122	042040																
11077	055604	045523	042101	020122																
11078	055612	041440	046131	042101																
11079	055620	024122	050122	031460																
11080	055626	020051	050040	054510																
11081	055634	020123	052502	040523																
11082	055642	051104	000200																	
11083	055646	041440	030523	020040	MSG13:	.ASCIZ	/CS1	WRDCNT	BUSADR	BADREX	DSKADR	CS2	CS3	DRVSTA	ERRREG/					
11084	055654	020040	051127	041504																
11085	055662	052116	020040	052502																
11086	055670	040523	051104	020040																
11087	055676	040502	051104	054105																
11088	055704	020040	051504	040513																
11089	055712	051104	020040	041440																
11090	055720	031123	020040	020040																
11091	055726	041440	031523	020040																
11092	055734	020040	051104	051526																
11093	055742	040524	020040	051105																
11094	055750	051122	043505	000200																
11095	055756	042504	041523	046131	MSG14:	.ASCIZ	/DESCYL	ER2	ER3	RPCC/<CRLF>										
11096	055764	020040	042440	031122																
11097	055772	020040	020040	042440																
11098	056000	031522	020040	020040																
11099	056006	050122	041503	000200																
11100	056014	040515	051523	041040	MSG15:	.ASCIZ	/MASS BUS TESTER /													
11101	056022	051525	052040	051505																
11102	056030	042524	020122	000																
11103	056035	040	051503	020061	MSG16:	.ASCIZ	/CS1	WRDCNT	BUSADR	BADREX	MR2	CS2	ST	ER	CS3/<					
11104	056042	020040	053440	042122																
11105	056050	047103	020124	041040																
11106	056056	051525	042101	020122																
11107	056064	041040	042101	042522																
11108	056072	020130	020040	046440																
11109	056100	031122	020040	020040																
11110	056106	041440	031123	020040																
11111	056114	020040	020040	052123																
11112	056122	020040	020040	042440																
11113	056130	020122	020040	020040																
11114	056136	041440	031523	000200																
11115	056144	020040	041503	020040	MSG17:	.ASCIZ	/CC	BUSADR	CR2	CR1	PHYS	BUSADR/<CRLF>								

K02

11116	056152	020040	052502	040523	
11117	056160	051104	020040	020040	
11118	056166	051103	020062	020040	
11119	056174	020040	051103	020061	
11120	056202	050040	054510	020123	
11121	056210	052502	040523	051104	
11122	056216	000200			
11123	056220	044124	020105	052521	MSG20: .ASCIZ /THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 0123456789/<15><12>
11124	056226	041511	020113	051102	
11125	056234	053517	020116	047506	
11126	056242	020130	052512	050115	
11127	056250	042105	047440	042526	
11128	056256	020122	044124	020105	
11129	056264	040514	054532	042040	
11130	056272	043517	020123	040502	
11131	056300	045503	030040	031061	
11132	056306	032063	033065	034067	
11133	056314	006471	000012		
11134	056320	046111	042514	040507	MSG21: .ASCIZ /ILLEGAL DEVICE/<CRLF>
11135	056326	020114	042504	044526	
11136	056334	042503	000200		
11137	056340	020040	020040	020040	MSG22: .ASCII / /
11138	056346	020040			
11139	056350	020040	020040	020040	MSG23: .ASCIZ / /
11140	056356	000			
11141	056357	125	044516	052502	MSG24: .ASCIZ /UNIBUS EXERCISER /
11142	056364	020123	054105	051105	
11143	056372	044503	042523	020122	
11144	056400	000			
11145	056401	117	052120	041456	MSG25: .ASCIZ /OPT.CP=/
11146	056406	036520	000		
11147	056411	125	042516	050130	EM1: .ASCIZ /UNEXPECTED TRAP TO 4/
11148	056416	041505	042524	020104	
11149	056424	051124	050101	052040	
11150	056432	020117	000064		
11151	056436	044526	052122	041520	DH1: .ASCIZ /VIRTPC PHYSPC PSW CPUERR/
11152	056444	020040	044120	051531	
11153	056452	041520	020040	020040	
11154	056460	051520	020127	020040	
11155	056466	050103	042525	051122	
11156	056474	000			
11157	056475	000	001	000	DF1: .BYTE 0,1,0,0,0
11158	056500	000	000		
11159					.EVEN
11160	056502	001536	001536	001736	DT1: .WORD VADR,VADR,REPPSW,REPREG,0
11161	056510	001740	000000		
11162	056514	047125	054105	042520	EM2: .ASCIZ /UNEXPECTED TRAP TO 10/
11163	056522	052103	042105	052040	
11164	056530	040522	020120	047524	
11165	056536	030440	000060		
11166	056542	044526	052122	041520	DH2: .ASCIZ /VIRTPC PHYSPC PSW/
11167	056550	020040	044120	051531	
11168	056556	041520	020040	020040	
11169	056564	051520	000127		
11170					.EVEN
11171	056570	001536	001536	001736	DT2: .WORD VADR,VADR,REPPSW,0



11172	056576	000000							
11173	056600	047125	054105	042520	EM3:	.ASCIZ	/UNEXPECTED TRAP TO 250(MGMT)/		
11174	056606	052103	042105	052040					
11175	056614	040522	020120	047524					
11176	056622	031040	030065	046450					
11177	056630	046507	024524	000					
11178	056635	126	051111	050124	DH3:	.ASCIZ	/VIRTPC PHYSPC PSW MMRO MMR2/		
11179	056642	020103	050040	054510					
11180	056650	050123	020103	020040					
11181	056656	050040	053523	020040					
11182	056664	020040	046515	030122					
11183	056672	020040	020040	046515					
11184	056700	031122	000						
11185		056704							
11186	056704	001536	001536	001736	DT3:	.EVEN			
11187	056712	001744	001746	000000		.WORD	VADR, VADR, REPPSW, REPMRO, REPMR2, 0		
11188	056720	047125	054105	042520	EM4:	.ASCIZ	/UNEXPECTED TRAP TO 114/		
11189	056726	052103	042105	052040					
11190	056734	040522	020120	047524					
11191	056742	030440	032061	000					
11192	056747	126	051111	050124	DH4:	.ASCIZ	/VIRTPC PHYSC PC PSW MEM ERR REG/		
11193	056754	020103	050040	054510					
11194	056762	041523	050040	020103					
11195	056770	050040	053523	020040					
11196	056776	046440	046505	042440					
11197	057004	051122	051040	043505					
11198	057012	000							
11199		057014							
11200	057014	001536	001536	001736	DT4:	.EVEN			
11201	057022	001740	000000			.WORD	VADR, VADR, REPPSW, REPREG, 0		
11202	057026	000	001	000	DF4:	.BYTE	0, 1, 0, 0		
11203	057031	000							
11204	057032	040520	044522	054524	EM5:	.ASCIZ	/PARITY ERROR DURING DATA CHECK/		
11205	057040	042440	051122	051117					
11206	057046	042040	051125	047111					
11207	057054	020107	040504	040524					
11208	057062	041440	042510	045503					
11209	057070	000							
11210	057071	123	041522	042101	DH5:	.ASCIZ	/SRCADR DSTADR MEM ERR REG/		
11211	057076	020122	042040	052123					
11212	057104	042101	020122	046440					
11213	057112	046505	042440	051122					
11214	057120	051040	043505	000					
11215	057125	000	001	000	DF5:	.BYTE	0, 1, 0		
11216						.EVEN			
11217	057130	001750	001536	001740	DT5:	.WORD	SRCADR, VADR, REPREG, 0		
11218	057136	000000							
11219	057140	051105	047522	020122	EM6:	.ASCIZ	/ERROR DURING DATA CHECK-RELOC WAS BY CP/		
11220	057146	052504	044522	043516					
11221	057154	042040	052101	020101					
11222	057162	044103	041505	026513					
11223	057170	042522	047514	020103					
11224	057176	040527	020123	054502					
11225	057204	041440	000120						
11226	057210	051123	040503	051104	DH6:	.ASCIZ	/SRCADR DSTADR/		
11227	057216	020040	051504	040524					

M02

11228	057224	051104	000						
11229		057230				.EVEN			
11230	057230	001304	001536	000000	DT6:	.WORD	\$TMP0,VADR,0		
11231	057236	051105	047522	020122	EM10:	.ASCIZ	?ERROR DURING DATA CHECK-RELOC WAS BY I/O?		
11232	057244	052504	044522	043516					
11233	057252	042040	052101	020101					
11234	057260	044103	041505	026513					
11235	057266	042522	047514	020103					
11236	057274	040527	020123	054502					
11237	057302	044440	047457	000					
11238	057307	123	041522	042101	DH10:	.ASCIZ	/SRCADR DSTADR DEVICE THAT DID XFER/		
11239	057314	020122	020040	051504					
11240	057322	040524	051104	020040					
11241	057330	042040	053105	041511					
11242	057336	020105	044124	052101					
11243	057344	042040	042111	054040					
11244	057352	042506	000122						
11245	057356	000	001	003	DF10:	.BYTE	0,1,3,0		
11246	057361	000							
11247						.EVEN			
11248	057362	001304	001536	001310	DT10:	.WORD	\$TMP0,VADR,\$TMP2,\$TMP3,0		
11249	057370	001312	000000						
11250	057374	020040	020040	020040	DH11:	.ASCIZ	/ DATA1 DATA2/		
11251	057402	020040	020040	020040					
11252	057410	020040	042040	052101					
11253	057416	030501	020040	020040					
11254	057424	020040	020040	020040					
11255	057432	020040	020040	020040					
11256	057440	020040	020040	020040					
11257	057446	020040	020040	020040					
11258	057454	020040	042040	052101					
11259	057462	031101	000						
11260	057465	005	000	005	DF11:	.BYTE	5,0,5,0		
11261	057470	000							
11262		057472				.EVEN			
11263	057472	001464	001450	001474	DT11:	.WORD	FLTMP0,FREG2,FLTMP1,FREG3,0		
11264	057500	001452	000000						
11265	057504	041125	020105	047516	EM12:	.ASCIZ	/UBE NON-EXISTANT MEMORY ERROR/		
11266	057512	026516	054105	051511					
11267	057520	040524	052116	046440					
11268	057526	046505	051117	020131					
11269	057534	051105	047522	000122					
11270	057542	044120	051531	041040	DH12:	.ASCIZ	/PHYS BUSADR/		
11271	057550	051525	042101	000122					
11272	057556	002			DF12:	.BYTE	2		
11273		057560				.EVEN			
11274	057560	001226	000000		DT12:	.WORD	\$GDDAT,0		
11275	057564	041115	020124	047516	EM13:	.ASCIZ	/MBT NON-EXISTANT MEMORY ERROR/		
11276	057572	026516	054105	051511					
11277	057600	040524	052116	046440					
11278	057606	046505	051117	020131					
11279	057614	051105	047522	000122					
11280	057622	044120	051531	040440	DH13:	.ASCIZ	/PHYS ADDRESS/		
11281	057630	042104	042522	051523					
11282	057636	000							
11283	057637	106	047514	052101	EM14:	.ASCIZ	/FLOATING POINT ERROR/		



```

11284 057644 047111 020107 047520
11285 057652 047111 020124 051105
11286 057660 047522 000122
11287 057664 042011 052101 030501
11288 057672 004411 004411 040504
11289 057700 040524 000062
11290
11291 057704 001434 001450 001440
11292 057712 001452 000000
11293 057716 004 000 004
11294 057721 000
11295 057722 042504 044526 042503
11296 057730 044040 047125 000107
11297
11298
11299 057736 000000
11300
11301
11302 057740 050117 051105 052101
11303 057746 047511 040516 020114
11304 057754 053523 052111 044103
11305 057762 051440 052105 044524
11306 057770 043516 100123
11307 057774 053523 052111 044103
11308 060002 004411 052411 042523
11309 060010 200
11310 060011 040 030440 020065
11311 060016 036440 030440 030060
11312 060024 030060 004460 044011
11313 060032 046101 020124 047117
11314 060040 042440 051122 051117
11315 060046 200
11316 060047 040 030440 020064
11317 060054 036440 030040 030064
11318 060062 030060 004460 046011
11319 060070 047517 020120 047117
11320 060076 052040 051505 100124
11321 060104 020040 031461 020040
11322 060112 020075 031060 030060
11323 060120 030060 004411 047111
11324 060126 044510 044502 020124
11325 060134 051105 047522 020122
11326 060142 054524 042520 052517
11327 060150 051524 200
11328 060153 040 030440 020062
11329 060160 036440 030040 030061
11330 060166 030060 004460 044411
11331 060174 044116 041111 052111
11332 060202 052440 042502 200
11333 060207 040 030440 020061
11334 060214 036440 030040 032060
11335 060222 030060 004460 044411
11336 060230 044116 041111 052111
11337 060236 044440 052124 051105
11338 060244 052101 047511 051516
11339 060252 200

```

```

DH14: .ASCIZ / DATA1 DATA2/
DT14: .EVEN
.WORD FTMP4,FREG2,FTMP6,FREG3,0
DF14: .BYTE 4,0,4,0
EM15: .ASCIZ /DEVICE HUNG/
.EVEN

```

```

ENDTAG: .WORD 0
:*****
:THE FOLLOWING ASCII GETS OVERLAYED WHEN THE PROGRAM RUNS.
SWITCH: .ASCII /OPERATIONAL SWITCH SETTINGS/<CRLF>

```

```

.ASCII /SWITCH USE/<CRLF>
.ASCII / 15 = 100000 HALT ON ERROR/<CRLF>
.ASCII / 14 = 040000 LOOP ON TEST/<CRLF>
.ASCII / 13 = 020000 INHIBIT ERROR TYPEOUTS/<CRLF>
.ASCII / 12 = 010000 INHIBIT UBE/<CRLF>
.ASCII / 11 = 004000 INHIBIT ITERATIONS/<CRLF>

```

11340	060253	040	030440	020060	.ASCII / 10 = 002000	BELL ON ERROR/<CRLF>
11341	060260	036440	030040	031060		
11342	060266	030060	004460	041011		
11343	060274	046105	020114	047117		
11344	060302	042440	051122	051117		
11345	060310	200				
11346	060311	040	020040	020071	.ASCII / 9 = 001000	LOOP ON ERROR/<CRLF>
11347	060316	036440	030040	030460		
11348	060324	030060	004460	046011		
11349	060332	047517	020120	047117		
11350	060340	042440	051122	051117		
11351	060346	200				
11352	060347	040	020040	020070	.ASCII ? 8 = 000400	INHIBIT RELOCATION VIA I/O DEVICE?<CRLF>
11353	060354	036440	030040	030060		
11354	060362	030064	004460	044411		
11355	060370	044116	041111	052111		
11356	060376	051040	046105	041517		
11357	060404	052101	047511	020116		
11358	060412	044526	020101	027511		
11359	060420	020117	042504	044526		
11360	060426	042503	200			
11361	060431	040	020040	020067	.ASCII / 7 = 000200	INHIBIT TYPEOUT OF THIS TEXT AND SYS SIZE/<CRLF>
11362	060436	036440	030040	030060		
11363	060444	030062	004460	044411		
11364	060452	044116	041111	052111		
11365	060460	052040	050131	047505		
11366	060466	052125	047440	020106		
11367	060474	044124	051511	052040		
11368	060502	054105	020124	047101		
11369	060510	020104	054523	020123		
11370	060516	044523	042532	200		
11371	060523	040	020040	020066	.ASCII / 6 = 000100	INHIBIT RELOCATION/<CRLF>
11372	060530	036440	030040	030060		
11373	060536	030061	004460	044411		
11374	060544	044116	041111	052111		
11375	060552	051040	046105	041517		
11376	060560	052101	047511	100116		
11377	060566	020040	032440	020040	.ASCII / 5 = 000040	INHIBIT ROUND ROBIN RELOCATION/<CRLF>
11378	060574	020075	030060	030060		
11379	060602	030064	004411	047111		
11380	060610	044510	044502	020124		
11381	060616	047522	047125	020104		
11382	060624	047522	044502	020116		
11383	060632	042522	047514	040503		
11384	060640	044524	047117	200		
11385	060645	040	020040	020064	.ASCII / 4 = 000020	INHIBIT RANDOM DISK ADDRESS/<CRLF>
11386	060652	036440	030040	030060		
11387	060660	031060	004460	044411		
11388	060666	044116	041111	052111		
11389	060674	051040	047101	047504		
11390	060702	020115	044504	045523		
11391	060710	040440	042104	042522		
11392	060716	051523	200			
11393	060721	040	020040	020063	.ASCII / 3 = 000010	INHIBIT MBT/<CRLF>
11394	060726	036440	030040	030060		
11395	060734	030460	004460	044411		



11396	060742	044116	041111	052111		
11397	060750	046440	052102	200		
11398	060755	040	020040	004462	.ASCII / 2	THESE THREE SWITCHES/<CRLF>
11399	060762	052011	042510	042523		
11400	060770	052040	051110	042505		
11401	060776	051440	044527	041524		
11402	061004	042510	100123			
11403	061010	020040	030440	004411	.ASCII / 1	ARE ENCODED TO SELECT RELOCATION/<CRLF>
11404	061016	051101	020105	047105		
11405	061024	047503	042504	020104		
11406	061032	047524	051440	046105		
11407	061040	041505	020124	042522		
11408	061046	047514	040503	044524		
11409	061054	047117	200			
11410	061057	040	020040	004460	.ASCII / 0	ON THE FOLLOWING DEVICES:/<CRLF>
11411	061064	047411	020116	044124		
11412	061072	020105	047506	046114		
11413	061100	053517	047111	020107		
11414	061106	042504	044526	042503		
11415	061114	035123	200			
11416	061117	011	027060	027056	.ASCII ?	0...RP11/RP03?<CRLF>
11417	061124	050122	030461	051057		
11418	061132	030120	100063			
11419	061136	030411	027056	051056	.ASCII ?	1...RK11/RK05?<CRLF>
11420	061144	030513	027461	045522		
11421	061152	032460	200			
11422	061155	011	027062	027056	.ASCII ?	2...NOT USED?<CRLF>
11423	061162	047516	020124	051525		
11424	061170	042105	200			
11425	061173	011	027063	027056	.ASCII ?	3...NOT USED?<CRLF>
11426	061200	047516	020124	051525		
11427	061206	042105	200			
11428	061211	011	027064	027056	.ASCII ?	4...RH11/RP04?<CRLF>
11429	061216	044122	030461	051057		
11430	061224	030120	100064			
11431	061230	032411	027056	051056	.ASCII ?	5...RH11/RS04 OR RS03?<CRLF>
11432	061236	030510	027461	051522		
11433	061244	032060	047440	020122		
11434	061252	051522	031460	200		
11435	061257	011	027066	027056	.ASCII ?	6...NOT USED?<CRLF>
11436	061264	047516	020124	051525		
11437	061272	042105	200			
11438	061275	011	027067	027056	.ASCIZ ?	7...NOT USED?<CRLF>
11439	061302	047516	020124	051525		
11440	061310	042105	000200			
11441		000001			.END	

AA	001547	1565#	2123#			
ADC82	011336	3311	3313#			
ADC85	012162	3543	3545#			
ADC86	012664	3701	3702	3704#		
ADC87	013614	3927	3928	3929	3931#	
ADC0	007230	2705	2706	2707	2709#	
ADC1	010130	2936	2937	2938	2940#	
ADC2	011140	3240	3242#			
ADC5	011762	3470	3471	3473#		
ADC6	012466	3643	3644	3646#		
ADC7	013462	3884	3885	3887#		
ADD0	014310	4095	4096	4097	4099#	
ADD1	014414	4133	4134	4136#		
ADD1A	014640	4216	4217	4218	4220#	
ADD1B	014656	4225	4226	4228#		
ADD2	015300	4372	4373	4375#		
ADD3	016056	4566	4568#			
ADD6	016434	4669	4670	4672#		
ADD7	017114	4788	4789	4790	4792#	
ARBEX	035606	8040	8084#			
ARBF IN	035574	8025	8047	8069	8071	8082#
ASHCLD	025332	6211#				
ASHCRD	025410	6232#				
ASHLD	025122	6161#				
ASHL1	025720	6345#				
ASHRD	025236	6188#				
ASHR1	026006	6368#				
ASLB1	010500	3071	3072	3074#		
ASLB1A	010724	3158	3159	3161#		
ASLB3	012152	3537	3538	3540#		
ASLB4	011442	3348	3349	3350	3352#	
ASLB6	012646	3693	3694	3695	3697#	
ASLB7	013712	3959	3960	3962#		
ASLD	007352	2749	2750	2751	2752	2754#
ASL1	010304	2999	3000	3001	3003#	
ASL3	011676	3439	3440	3442#		
ASL4	011232	3272	3273	3274	3276#	
ASL6	012436	3631	3632	3634#		
ASL7	013310	3831	3832	3834#		
ASP1	002502	1850#				
ASP2	002574	1880#				
ASRB1	010574	3107	3109#			
ASRB1A	010610	3113	3114	3116#		
ASRB2	011406	3333	3334	3336#		
ASRB2A	011424	3341	3342	3344#		
ASRB5	012112	3518	3519	3521#		
ASRB6	012764	3733	3734	3736#		
ASRB7	013730	3966	3967	3969#		
ASRD	007400	2763	2764	2765	2767#	
ASR1	010172	2956	2957	2958	2960#	
ASR2	011154	3246	3247	3249#		
ASR3	011662	3433	3435#			
ASR6	012320	3593	3594	3596#		
ASR7	013344	3845	3846	3848#		
BICB1	015040	4289	4290	4292#		
BICB1A	015062	4300	4303#	4362	4363	











# H03

DM3	056635	2048	11178#																	
DM4	056747	2053	11192#																	
DM5	057071	2058	11210#																	
DM6	057210	2063	11226#																	
DISPLA	001244	1493#	2203#	2211*	2480*	9583*	9605*													
DISPRE	000174	1450#	2211																	
DIVO	025634	6316#																		
DIV1	026102	6400#	6413																	
DPTRP =	000001	1437#	6776	6987	7001	7016														
DREG	002750	1962#																		
DSMR =	177570	1186#	1492	2202																
DT1	056502	2039	11160#																	
DT10	057362	2074	11248#																	
DT11	057472	2079	11263#																	
DT12	057560	2084	2089	11274#																
DT14	057704	2094	11291#																	
DT2	056570	2044	11171#																	
DT3	056704	2049	11186#																	
DT4	057014	2054	11200#																	
DT5	057130	2059	11217#																	
DT6	057230	2064	11230#																	
DUMMY	035412	8033#																		
DWORRY	041772	8775#																		
ECHO	054332	10814#	10832																	
EISOPT=	040000	1396#																		
EMTVEC=	000030	1275#	2186#	2187*	5299*	5301*	5338*	5339*												
EMT1	021314	5298	5306#																	
EMT1B	021374	5306	5310	5311	5314	5315	5319	5320	5323	5324	5325	5330#								
EMT1C	021400	5304	5332#																	
EMT1D	021412	5333	5334	5337#																
EM1	056411	2037	11147#																	
EM10	057236	2072	11231#																	
EM12	057504	2082	11265#																	
EM13	057564	2087	11275#																	
EM14	057637	2077	2092	11283#																
EM15	057722	2097	11295#																	
EM2	056514	2042	11162#																	
EM3	056600	2047	11173#																	
EM4	056720	2052	11188#																	
EM5	057032	2057	11204#																	
EM6	057140	2062	11219#																	
END	040640	8588#																		
ENDCP	031524	7259	7329#																	
ENDM	040606	8188	8572#																	
ENDMEM	040604	8227	8571#																	
ENDTAG	057736	2147	8226#	11299#																
END1	040640	8589#																		
ENTER2	037046	8278	8311#	8363	8498															
ERFLG	055066	10973#	10989#																	
ERPRT	055064	2143	2306	2462	5027	5508	6474	6662	6695	6733	6880	6907	6931	6955						
		7090	7142	7173	7186	7209	7224	7312	7425	8234	8589	10972#								
ERRADR	001734	1605#	9417#	9429	9478#	9488														
ERRBA	001726	1603#																		
ERRRTN	001340	1527#																		
ERRVEC=	000004	1268#	2138#	2143*	2200	2201*	2212*	2229*	2230*	2306*	2329*	2354*	2357*	2363*						
		2383#	2403#	2460	4985*	4986*	4987	5027*	5422*	5423*	5465*	5508*	5509*	5510*						





FREG2	001450	1546#	7623*	7675	7705*	7736	7754*	7793*	7845	7875*	7906	7924*	11263	11291
FREG3	001452	1547#	7663*	7677	7726*	7738	7758*	7833*	7847	7896*	7908	7928*	11263	11291
FREG4	001454	1548#												
FREG5	001456	1549#												
FREG6	001460	1550#												
FREG7	001462	1551#												
FRSTAD	001554	1568#	2495*	2614*	3790*	4442	4858*	5405*	5745*	6151*	7350*	7602*	8254	
FRSTNE	001556	1569#	2147*	8264										
FTMPO	001424	1536#	7614	7628	7687	7695	7715	7747	7755	7784	7798	7857	7865	7885
		7917	7925	10472	10487	10519								
FTMP1	001426	1537#												
FTMP2	001430	1538#	7615	7636	7645	7688	7697	7707	7748					
FTMP3	001432	1539#												
FTMP4	001434	1540#	7622*	7674	7704*	7735	7753*	7785	7806	7815	7858	7867	7877	7918
		11291												
FTMP5	001436	1541#												
FTMP6	001440	1542#	7662*	7725*	7756*	11291								
FTMP7	001442	1543#												
GNS	= ***** U	1449	2317	2451	6618	8361	8370	8646	8653	9807	10633	10634	10635	10636
		10637	10640	10641	10642	10643								
GNMHAM	002542	1867#												
GSTST	007700	2870#												
HALT1	031412	7301#												
HI	= 000200	1440#												
HITMIS	= 177752	1343#												
HT	= 000011	1178#	9929	9972										
INC81	010442	3053	3054	3056#										
INC82	011550	3390	3391	3392	3394#									
INC83	012122	3524	3526#											
INC86	012630	3688	3690#											
INC86A	013000	3739	3740	3742#										
INC87	013662	3947	3948	3950#										
INC0	007312	2731	2732	2733	2734	2736#								
INC1	010224	2971	2972	2973	2975#									
INC3	011750	3465	3467#											
INC4	011072	3216	3217	3219#										
INC6	012414	3622	3623	3624	3626#									
INC7	013474	3890	3892#											
INIT	002757	1969#												
IOMON	036770	8235	8300#	8499										
IOTTST	021040	5232#												
IOTVEC	= 000020	1273#	2184*	2185*	5232	5245*	5294*	5295*	5340*	5341*	7274*	7293*		
IOWC	001610	1582#	8277*	8310*	8401									
JAM	002742	1956#												
JMP1	020254	5043	5044	5045	5047#									
JMP3	020324	5063	5067#											
JMP4	020362	5079	5083#											
JMP5	020426	5097	5101#											
JMP6	020446	5109#												
JMP7	020504	5120#												
JSR1	020516	5127#												
JSR3	020602	5137	5151#											
JSR4	020670	5163	5176#											
JSR6	020746	5185	5199#											
JSR7	021000	5203	5213#											
KERSTK	= 001200	1351#	2105	2150	5499	8562	8592							



# K03

MAINDEC-11-DQKDC-A PDP 11/6X SERIES CPU EXERCISER MACY11 27(1006) 07-FEB-77 10:08 PAGE 244  
 DQKDC.A.P11 07-FEB-77 09:58 CROSS REFERENCE TABLE -- USER SYMBOLS

KIPAR0=	172340	1328#	7517	8103#	8202#	8551#	8558	10693	10887	10912
KIPAR1=	172342	1329#	8104#	8203#	8552#					
KIPAR2=	172344	1330#	8105#	8204#	8553#					
KIPAR3=	172346	1331#	8113#	8121#	8130	8205#	8206	8208	8302	8551
KIPAR4=	172350	1332#	7532#	8206#	8207#	8552				
KIPAR5=	172352	1333#	8208#	8209#	8553					
KIPAR6=	172354	1334#								
KIPAR7=	172356	1335#	8106#	8210#	8222					
KIPDR0=	172300	1317#	7513	8096#						
KIPDR1=	172302	1318#	8097#							
KIPDR2=	172304	1319#	8098#							
KIPDR3=	172306	1320#	8099#							
KIPDR4=	172310	1321#	7533#	8100#						
KIPDR5=	172312	1322#	8101#							
KIPDR6=	172314	1323#								
KIPDR7=	172316	1324#	8102#							
KM	= 000000	1432#								
KTABRT	054662	2478	7292	10924#						
KTABT	032364	7530#								
KTEX	032570	7531	7574#							
KTFLG	054664	10925#	10942#							
KTOPT	= 100000	1395#								
KTPAR	032226	7477#								
KTPDR	032110	7438#								
LCDTA	002734	1950#								
LCUA	002674	1918#								
LDKT	054516	9378	9447	9513	9980	10804	10882#			
LDRP3	041612	8730	8738#	8997						
LDRP4	042514	8866	8873#	9194	9220	9233	9248	9268		
LDRS	043044	8927	8936#	9318	9346					
LF	= 000012	1179#	9966	9972						
LFGIN	002676	1919#								
LJAM	002666	1915#								
LKOPT	= 001000	1399#	2238	8046	8140					
LKS	= 177546	1364#	2236	8056#	8058	8144#	9525#			
LKSRV	046630	8078	8142	9512#						
LKVEC	= 000100	1365#	8049#	8050#	8078#	8142#	8143#			
LO	= 000100	1439#								
LOOP	005530	2437	2439	2441	2443	2445	2447	2458#	8564	8611
LPBA	002672	1917#								8669
LSERV	002670	1916#								
LSTMEM	001560	1570#	2145#	2146#	8266					
LTAG	002702	1921#								
LTICKS	001644	1596#	8426	8443	9514#	9515	9517#	9518#	9519	9521#
LHAM	002700	1920#								9981
MARKEX	024440	6018	6022	6026#						
MARK1	024414	6014	6016#							
MBTAS	= 160116	1385#								
MBTBA	= 160104	1380#	1807							
MBTBAE	= 160174	1389#								
MBTCS1	= 160100	1378#	1805	1808						
MBTCS2	= 160110	1382#	1810							
MBTCS3	= 160176	1390#	1813							
MBTDB	= 160120	1386#								
MBTDT	= 160126	1388#	1816							
MBTER	= 160114	1384#	1812							































TST2	006714	2604#		
TST20	014346	4117#		
TST21	014672	4237#		
TST22	015222	4359#		
TST23	015506	4447#		
TST24	015712	4518#		
TST25	016102	4582#		
TST26	016262	4639#		
TST27	016560	4701#		
TST3	007116	2669#		
TST30	016734	4736	4744#	
TST31	017154	4809#		
TST32	017304	4848#		
TST33	017506	4903#		
TST34	020214	5032#		
TST35	020506	5124#		
TST36	021030	5229#		
TST37	021234	5247	5290#	
TST4	007434	2786#		
TST40	021442	5346#		
TST41	021652	5395#		
TST42	022350	5514#		
TST43	022514	5561#		
TST44	022756	5619#		
TST45	023164	5686#		
TST46	023316	5735#		
TST47	024116	5910#		
TST5	010030	2909#		
TST50	024346	6000#		
TST51	024442	6051#		
TST52	024642	6094#		
TST53	025022	6141#		
TST54	025474	6261#		
TST55	025624	6313#		
TST56	026072	6397#		
TST57	026150	6443#		
TST6	010362	3033#		
TST60	026330	6479#		
TST61	026364	6503#		
TST62	026540	6553#		
TST63	026610	6593#		
TST64	027076	6677#		
TST65	027324	6759#		
TST66	027564	6833#		
TST67	027702	6872#		
TST7	010770	3184#		
TST70	030074	6945#		
TST71	030164	6983#		
TST72	030550	7121#		
TST73	030762	7197#		
TST74	031212	7255#		
TST75	031402	7298#		
TST76	031466	7318#		
TST77	031542	7340#		
TOPT =	000400	1400#	2242	8023
TTYCHK	035342	8021#		





K04

UIPDR7=	177616	1302#	8216*																	
UJAM	002776	1984#																		
ULCOTA	002762	1978#																		
UM	= 140000	1433#	5259	5433	6101	6104	6125	7258	7393	7402										
UMD	002766	1980#																		
UNITNO	001616	1585#	8400*	8404	8409															
UNUA	003022	1994#																		
UPBA	003002	1986#																		
URES	003024	1995#																		
UREV	003014	1991#																		
UR6A	002532	1863#																		
UR6B	002626	1896#																		
USERV	003000	1985#																		
USESTK=	000700	1337#	2458	5026	5286	5458	5493	5507	6265	6456	7251	7294	7399	7419						
		7423	10983																	
USREG	003016	1992#																		
UM6	012244	3576#	3580*	3587*	3592*	3599*	3607*	3614*	3621*	3628*	3630*	3636*	3642*	3648*						
UM7	013210	3802#	3906	3909*	3914*															
UM7	013214	3807	3805#																	
VADR	001536	1559#	6708*	8286*	8466*	8544*	9671	9672*	9678*	9680	9682*	10685	10734*	10864*						
		10865*	10933*	10934*	10959*	10960*	10981*	10982*	11160	11171	11186	11200	11217	11230						
		11248																		
WCNSSM=	000226	1407#	10581																	
WCSADR	002636	1900#																		
WCSA.0	002536	1865#																		
WCSA.1	002540	1866#																		
WCSB.0	002632	1898#																		
WCSB.1	002634	1899#																		
WRFLAG=	000344	1408#	6609	6643																
WRINIT=	000352	1425#	2216																	
WRLCUR=	000303	1416#																		
WRLDAT=	000306	1422#	6783																	
WRLFGI=	000304	1418#																		
WRLJAM=	000300	1410#																		
WRLPBA=	000302	1414#	6781																	
WRLSER=	000301	1412#																		
WRLTAG=	000307	1424#	6785																	
WRLJHA=	000305	1420#																		
WRJHAM=	000222	1406#	6604	6661	6689	6738	7000	7086	7129	7167	7191									
WMP	= 000100	1436#	6774	6838	6840	6992	6995													
X	054512	10872	10875#																	
XOR0	023472	5772	5773	5774	5777#															
XOR1	023560	5797	5798	5799	5800	5802	5805#													
XOR24	023614	5819#																		
XOR35	024064	5888	5889	5893#																
XOR6	023702	5824	5827	5829	5836#															
XOR6A	023706	5822#	5823#	5828#	5830#	5834	5840#	5845*	5856*	5858*										
XOR6B	023710	5831#	5832#	5833#	5834	5841#														
XOR7	024114	5904#																		
XXDP	003324	2113#	2126#	2134																
XXDPC	003325	2114#	2129#	2438																
SACO	001410	1530#	7617#	7620#	7627*	7629	7638*	7639	7648*	7652*	7656*	7660*	7675*	7690*						
		7698#	7702#	7708#	7710	7716*	7718	7723*	7736*	7750*	7757*	7787*	7790*	7797*						
		7799	7808#	7809	7818*	7822*	7826*	7830*	7845*	7860*	7868*	7872*	7878*	7880						
		7886#	7888	7893#	7906*	7920*	7927*	7944	7955	7969	7974	7985	7987	10543						
SAC1	001412	1531#	7616#	7620	7623	7629*	7633	7639*	7642	7649*	7663	7671	7677*	7680*						



		7681	7689*	7693	7696*	7705	7710*	7713	7718*	7720	7726	7732	7738*	7741*
		7742	7749*	7754	7760*	7762*	7763	7786*	7790	7793	7799*	7803	7809*	7812
		7819*	7833	7841	7847*	7850*	7851	7859*	7863	7866*	7875	7880*	7883	7888*
		7890	7896	7902	7908*	7911*	7912	7919*	7924	7930*	7932*	7933	7944*	7955*
		7969	7973	7986	7987*									
SAC2	001414	1532*	7633*	7660	7668	7670*	7671*	7672*	7673*	7681	7693*	7702	7713*	7723
		7729	7731*	7732*	7733*	7734*	7742	7803*	7830	7838	7840*	7841*	7842*	7843*
		7851	7863*	7872	7883*	7893	7899	7901*	7902*	7903*	7904*	7912		
SAC3	001416	1533*	7642*	7656	7668	7670	7720*	7729	7731	7812*	7826	7838	7840	7890*
		7899	7901											
SAC4	001420	1534*												
SAC5	001422	1535*												
SAUTO8	001236	1489*												
SBDADR	001224	1484*												
SBDAT	001230	1486*	9425*	9486*										
SBELL	001330	1522*	9608	9633										
SBLFF	001406	1529*	10398*	10423										
SCHARC	051004	9936*	9946*	9953	9964*	9969*								
SCKSMR=	***** U	10640												
SCHTAG	001200	1471*	2177	2178	2186	2192	2193	2194						
SCH1 =	000010	1504*	1505*	1506*	1507*	1508*	1509*	1510*	1511*	1512*				
SCH2 =	000020	1504*	1505*	1506*	1507*	1508*	1509*	1510*	1511*	1512*				
SCH3 =	000010	1502*	1504											
SCH4 =	000010	1512*	1513*	1514*	1515*	1516*	1517*	1518*	1519*	1520*				
SCRLF	001335	1524*	8658	9616	9633	9663	9666	9704	9719	9724	9728	9873	9889	9890
		9935	9972	10023	10043	10065	10780	10788	10798					
SDBLK	052050	10170	10204	10212*										
SDB20	052060	9684	9755	9868	10224*									
SDOAGN	041240	8640	8661	8667*										
SDTBL	052040	10173	10208*											
SENDAD	041230	1460	2117	2313	8663*	9628								
SENDCT	041074	2192	8642*											
SENULL	041244	8670*												
SEOP	041036	8604	8632*											
SEOPCT	041066	2192*	8639*	8643										
SERFLG	001204	1474*	9533	9555	9557	9563*	9580	9582	9587	9602	9603*	9633		
SERMAX	001217	1481*	2195*	9557	9579*	9587								
SERPSM	001576	1577*												
SERROR	047160	2186	5294	5338	7205	9601*								
SERRPC	001220	1482*	9610*	9611*	9612	9633	9667	9672						
SERRTB	003126	2035*	9714											
SERTY	047334	9615	9662*											
SERTTL	001214	1479*	8655	8659*	9609*	9633								
SESCAP	001326	1521*	2194*	9578*	9624	9626	9633							
SFACTO	001552	1567*	8260*	8376	8453	8456								
SFILLC	001260	1500*	9939	9972										
SFILLS	001257	1499*	9972											
SFLBUF	001342	1528*	10322	10370										
SFLD20	052600	10392*	10643											
SFL20	052312	10317*	10642											
SGDADR	001222	1483*												
SGDAT	001226	1485*	9424*	9485*	11274									
SGET42	041220	8660*												
SCTSAR=	***** U	10639												
SHD =	000000	1113												
SHINUM	001566	1573*	2106*	8689	8762	8832	8905	10443	10451	10456*	10479	10483		







	7123*	7199*	7257*	7300*	7320*	7343*	7347*	7436*	7476*	7529*	7595*	7599*	7780*
	8019*	8020	8091*	8092	8226	8560*	8634*	9533	9582*	9583	9587	9602*	9605
	9633	9693											
STYPBN= ***** U	10638												
STYPOS 051634	10158#	10637											
STYPE 050562	9917#	10625	10633										
STYPEC 050732	9938	9945	9952	9957#	9960								
STYPEX 051006	9958	9965	9967	9970#									
STYPOC 051432	10099#	10634											
STYPON 051446	10098	10101#	10636										
STYPOS 051406	10094#	10635											
SXTSTR 046730	9544#												
SSGET4= 000000	8662#												
SOFILL 051631	10095*	10099*	10109	10144#									
S4OCAT= ***** U	9541	9615											
. = 061314	1445#	1449#	1458	1459#	1461#	1463#	1470#	1475#	1526	1528#	1552#	1553#	1555#
	1598#	1599#	1600#	1601#	1602#	1603#	1726#	1826#	2181	2196	2197	2318#	2497
	2503	2616	2622	2633	2641	2649	2656	2664	2680	2690	2700	2708	2717
	2726	2735	2744	2753	2759	2766	2773	2781	2807	2817	2827	2834	2841
	2850	2866	2913	2923	2930	2939	2946	2952	2959	2966	2974	2981	2989
	2994	3002	3008	3016	3023	3028	3036	3049	3055	3061	3067	3073	3079
	3086	3093	3100	3108	3115	3119	3126	3131	3138	3145	3151	3155	3160
	3166	3170	3176	3179	3187	3199	3204	3211	3218	3224	3229	3233	3241
	3248	3255	3259	3265	3269	3275	3279	3282	3290	3304	3308	3312	3320
	3328	3335	3343	3351	3357	3363	3369	3374	3379	3385	3393	3397	3400
	3408	3421	3428	3434	3441	3449	3455	3461	3466	3472	3479	3485	3493
	3511	3514	3520	3525	3533	3539	3544	3551	3555	3561	3567	3584	3590
	3595	3603	3611	3617	3625	3633	3639	3645	3649	3652	3669	3674	3680
	3689	3696	3703	3710	3716	3723	3729	3735	3741	3748	3753	3759	3765
	3768	3792	3798	3818	3825	3833	3840	3847	3855	3862	3868	3875	3879
	3886	3891	3898	3922	3930	3936	3943	3949	3955	3961	3968	3974	3981
	3988	4000	4008	4014	4023	4031	4051	4059	4069	4077	4085	4088	4098
	4103	4112	4120	4135	4144	4152	4161	4168	4175	4181	4189	4200	4207
	4212	4219	4227	4232	4240	4258	4262	4269	4280	4285	4291	4296	4302
	4305	4354	4374	4381	4386	4393	4404	4414	4421	4427	4435	4440	4450
	4483	4491	4495	4501	4508	4513	4545	4553	4558	4561	4567	4572	4577
	4612	4615	4621	4628	4634	4642	4655	4660	4664	4671	4677	4684	4690
	4694	4718	4722	4725	4730	4734	4766	4769	4770	4776	4783	4791	4795
	4799	4803	4836	4860	4866	4872	4878	4883	4887	4893	4898	4940	4945
	4951	4955	4965	4975	4980	5016	5018#	5039	5041	5046	5051	5059	5061
	5066	5082	5086	5100	5108	5119	5207	5238	5254	5273	5280	5298	5328
	5329	5336	5352	5360	5363	5367	5372	5407	5413	5421	5495	5573	5588
	5644	5651	5659	5665	5676	5697	5712	5721	5747	5753	5763	5776	5787
	5790	5804	5818	5835	5838	5850	5859	5863	5870	5876	5881	5892	5898
	5903	5919	5938	5955	5959	5961	5964	5989	5995	6015	6020	6024	6058
	6111	6123	6153	6159	6182	6205	6225	6249	6273	6285	6304	6336	6360
	6384	6416	6458	6472	6490	6681	6683	7261	7273	7302	7309	7326	7352
	7359	7384	7411	7539	7547	7569	7604	7610	8362#	8548	8596	8647#	8670
	8671#	8810	8963	8975	9013	9048	9070	9083	9123	9158	9586	9587	9633
	9899#	9972	10212#	10254#	10570	10597	10820	10919#	11185#	11199#	11229#	11262#	11273#
.PARSR 054414	2476	6798	6859	7017	10729	10854#							



COMMEN	1280#														
ENDCOM	1280#														
ERROR	1174#	1430	5342	7683	7744	7766	7853	7914	7935	8288	8445	8495	8500	8545	9422
	9426	9483	9487	10736	10869	10939	10963	10987							
ESCAPE	1280#														
FLTST1	7582#	7584	7770												
GETPRI	1280#														
GETSMR	1280#	2316#													
HLT	1430#	2634	2642	2650	2657	2665	2681	2691	2701	2709	2718	2727	2736	2745	2754
	2760	2767	2774	2782	2808	2818	2828	2835	2842	2851	2867	2901	2924	2931	2940
	2947	2953	2960	2967	2975	2982	2990	2995	3003	3009	3017	3024	3029	3050	3056
	3062	3068	3074	3080	3087	3094	3101	3109	3116	3120	3127	3132	3139	3146	3152
	3156	3161	3167	3171	3177	3180	3200	3205	3212	3219	3225	3230	3234	3242	3249
	3256	3260	3266	3270	3276	3280	3283	3305	3309	3313	3321	3329	3336	3344	3352
	3358	3364	3370	3375	3380	3386	3394	3398	3401	3422	3429	3435	3442	3450	3456
	3462	3467	3473	3480	3486	3512	3515	3521	3526	3534	3540	3545	3552	3556	3562
	3568	3585	3591	3596	3604	3612	3618	3626	3634	3640	3646	3650	3653	3670	3675
	3681	3690	3697	3704	3711	3717	3724	3730	3736	3742	3749	3754	3760	3766	3819
	3826	3834	3841	3848	3856	3863	3869	3876	3880	3887	3892	3899	3923	3931	3937
	3944	3950	3956	3962	3969	3975	3982	3989	4001	4009	4015	4024	4032	4047	4052
	4060	4070	4078	4086	4089	4099	4104	4113	4136	4145	4153	4162	4169	4176	4182
	4190	4201	4208	4213	4220	4228	4233	4259	4270	4281	4286	4292	4297	4303	4355
	4375	4382	4387	4394	4405	4415	4422	4428	4436	4441	4484	4492	4496	4502	4509
	4514	4546	4554	4559	4562	4568	4573	4578	4613	4616	4622	4629	4635	4656	4661
	4666	4672	4678	4685	4691	4695	4719	4723	4726	4731	4735	4767	4771	4777	4784
	4792	4796	4800	4804	4873	4879	4884	4888	4894	4899	4941	4946	4952	4956	4966
	4976	4991	4993	4995	4997	4999	5001	5003	5005	5007	5009	5011	5013	5015	5017
	5019	5047	5067	5083	5101	5109	5120	5148	5173	5184	5196	5204	5208	5222	5246
	5281	5293#	5305	5330	5337	5342#	5362	5365	5369	5374	5459	5496	5526	5527	5528
	5574	5589	5594	5631	5645	5652	5660	5666	5677	5698	5713	5722	5764	5777	5791
	5805	5819	5836	5851	5860	5871	5877	5882	5893	5899	5904	5920	5939	5956	5960
	5965	5971	5975	5981	5990	5996	6017	6021	6025	6075	6080	6116	6127	6183	6206
	6226	6250	6286	6305	6337	6361	6385	6417	6465	6469	6491	6529	6568	6613	6629
	6649	6692	6699	6706	6729	6792	6820	6846	6856	6884	6892	6899	6910	6920	6928
	6951	6959	6966	7007	7022	7030	7040	7050	7069	7078	7095	7146	7159	7180	7214
	7229	7278	7290	7306	7327	7371	7400	7420	7465	7506	7548	7567	8070		
MSG	6668#	6671	6748#	6751	6825#	6828	6860#	6863	6934#	6937	6970#	6973	7105#	7108	
MSG8	4740#	4742													
MSG8	4805#	4807													
MSGC	5223#	5226													
MSGD	5905#	5907													
MSGE	6027#	6029													
MSGF	6255#	6257													
MSGG	6306#	6308													
MSGH	6391#	6393													
MSGJ	6436#	6438													
MSGK	7428#	7431													
MSGL	7469#	7472													
MSGM	7521#	7523													
MSGM1	6492#	6495													
MSGM10	6542#	6545													
MSGM11	6575#	6578													
MSGO	8084#	8087													
MULT	1280#														
MYTAGS	1464#	1527													
NEWTST	1280#	1464#	2482	2601	2666	2783	2906	3030	3181	3284	3402	3487	3569	3654	3777

	3900	3990	4114	4234	4356	4444	4515	4579	4636	4698	4740	4805	4845	4900	5029
	5121	5223	5287	5343	5392	5511	5558	5616	5683	5732	5905	5997	6027	6090	6138
	6255	6306	6391	6436	6476	6493	6543	6576	6669	6749	6826	6861	6935	6971	7106
	7194	7252	7295	7315	7337	7428	7469	7521	7582	7768	8014	8084			
POP	1280#	10199	10293	10457	10582	10583									
PUSH	1280#	10158	10273	10438	10560	10566									
RELOCA	2102#	2489	2608	3784	4852	5399	5739	6145	7344	7596					
RELOCB	2102#	2594	3770	4838	5385	5725	6131	7329	7575	8007					
REPORT	1280#														
SCOPE	1175#	2487	2594	2606	2670	2787	2910	3034	3185	3288	3406	3491	3573	3658	3770
	3782	3904	3994	4118	4238	4360	4448	4519	4583	4640	4702	4745	4810	4838	4850
	4904	5033	5125	5230	5291	5347	5385	5397	5515	5562	5620	5687	5725	5737	5911
	6001	6052	6095	6131	6143	6262	6314	6398	6444	6480	6504	6554	6594	6678	6760
	6834	6873	6946	6984	7122	7198	7256	7299	7319	7330	7342	7435	7475	7528	7575
	7594	7779	8007												
SETPRI	1280#														
SETTRA	10625#	10634	10635	10636	10637	10640	10641	10642	10643						
SETUP	1280#	2176													
SKIP	1280#	2437	2439	2441	2443	2445	2447	4736	5247	7510	7765				
SLASH	1280#														
SPACE	1280#														
STARS	1280#	1456	1466	1526	2104	2319	2328	2348	2360	2362	2382	2402	2482	2484	2601
	2603	2666	2668	2783	2785	2906	2908	3030	3032	3181	3183	3284	3286	3402	3404
	3487	3489	3569	3571	3654	3656	3777	3779	3900	3902	3990	3992	4114	4116	4234
	4236	4356	4358	4444	4446	4515	4517	4579	4581	4636	4638	4698	4700	4740	4743
	4805	4808	4845	4847	4900	4902	5029	5031	5121	5123	5224	5228	5287	5289	5343
	5345	5392	5394	5511	5513	5558	5560	5616	5618	5683	5685	5732	5734	5905	5909
	5997	5999	6027	6050	6091	6093	6138	6140	6255	6260	6306	6312	6391	6396	6436
	6442	6476	6478	6493	6502	6543	6552	6576	6592	6669	6676	6749	6758	6826	6832
	6861	6871	6935	6944	6971	6982	7106	7120	7194	7196	7252	7254	7295	7297	7315
	7317	7337	7339	7429	7433	7470	7473	7521	7526	7582	7591	7684	7746	7768	7777
	7854	7916	7938	7943	7950	7954	7961	7967	8014	8016	8085	8088	8146	8161	8180
	8185	8236	8249	8293	8299	8582	8587	8625	8672	8675	8744	8747	8814	8817	8888
	8891	8949	8952	9055	9058	9171	9174	9275	9278	9373	9376	9442	9445	9505	9511
	9530	9589	9633	9660	9741	9750	9762	9769	9778	9788	9798	9814	9822	9847	9880
	9902	9972	9978	10032	10038	10071	10148	10215	10257	10300	10316	10373	10391	10426	10461
	10469	10496	10513	10556	10572	10604	10644	10648	10667	10683	10708	10719	10740	10742	10747
	10749	10758	10771	10825	10829	10838	10853	10876	10881	10901	10904	10921	10923	10946	10948
	10969	10971	10995	11300											
SMRSU	1280#	2198#													
TRMTRP	10625#														
TYPBIN	1280#														
TYPDEC	1280#	8648	8655	9701											
TYPNAM	1104#	1280#	2309												
TYPNUM	1280#														
TYPOCS	1280#														
TYPOCT	1280#	9667													
TYPTXT	1280#	2449	6616	8359	8368	8644	8651	9805							
UPCODE	1464#	10579													
SSCMRE	1464#	1504	1505	1506	1507	1508	1509	1510	1511						
SSCMTM	1464#	1512	1513	1514	1515	1516	1517	1518	1519						
SSESCA	1280#														
SSNEWT	1280#	1464#	2482	2601	2666	2783	2906	3030	3181	3284	3402	3487	3569	3654	3777
	3900	3990	4114	4234	4356	4444	4515	4579	4636	4698	4740	4805	4845	4900	5029
	5121	5224	5287	5343	5392	5511	5558	5616	5683	5732	5905	5997	6027	6091	6138
	6255	6306	6391	6436	6476	6493	6543	6576	6669	6749	6826	6861	6935	6971	7106



E05

SSSET	7194	7252	7295	7315	7337	7429	7470	7521	7582	7768	8014	8085
SSSKIP	10625#	10634	10635	10636	10637	10640	10641	10642	10643			
.EQUAT	1280#	4736	5247	7511	7765							
.HEADE	1104#	1170										
.KT11	1104#	1280										
.SETUP	1104#	2157										
.SMRHI	1104#	1123										
.SMRLO	1104#	1135#	1138	1141								
.SACT1	1104#	1454										
.SCATC	1104#	1443										
.SCMTA	1464#											
.SDB20	1464#	10213										
.SEOP	1104#	8623										
.SERRO	1104#	9587										
.SPOWE	1104#	10554										
.SRAND	1464#	10426										
.SRDOC	1104#											
.SREAD	1104#											
.SSAVE	1104#	10255										
.SSCOP	1464#	9528										
.STRAP	1104#	10602										
.STYPD	1104#	10146										
.STYPE	1464#	9900										
.STYPO	1104#	10069										

. ABS. 061314 000

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

DSKZ:DQKDC.A, DSKZ:DQKDC.A, SEQ/CRF/SOL/DS:ERFZ=DSKZ:DQKDC.A.P11  
RUN-TIME: 27 31 3 SECONDS  
RUN-TIME RATIO: 680/63=10.7  
CORE USED: 31K (62 PAGES)