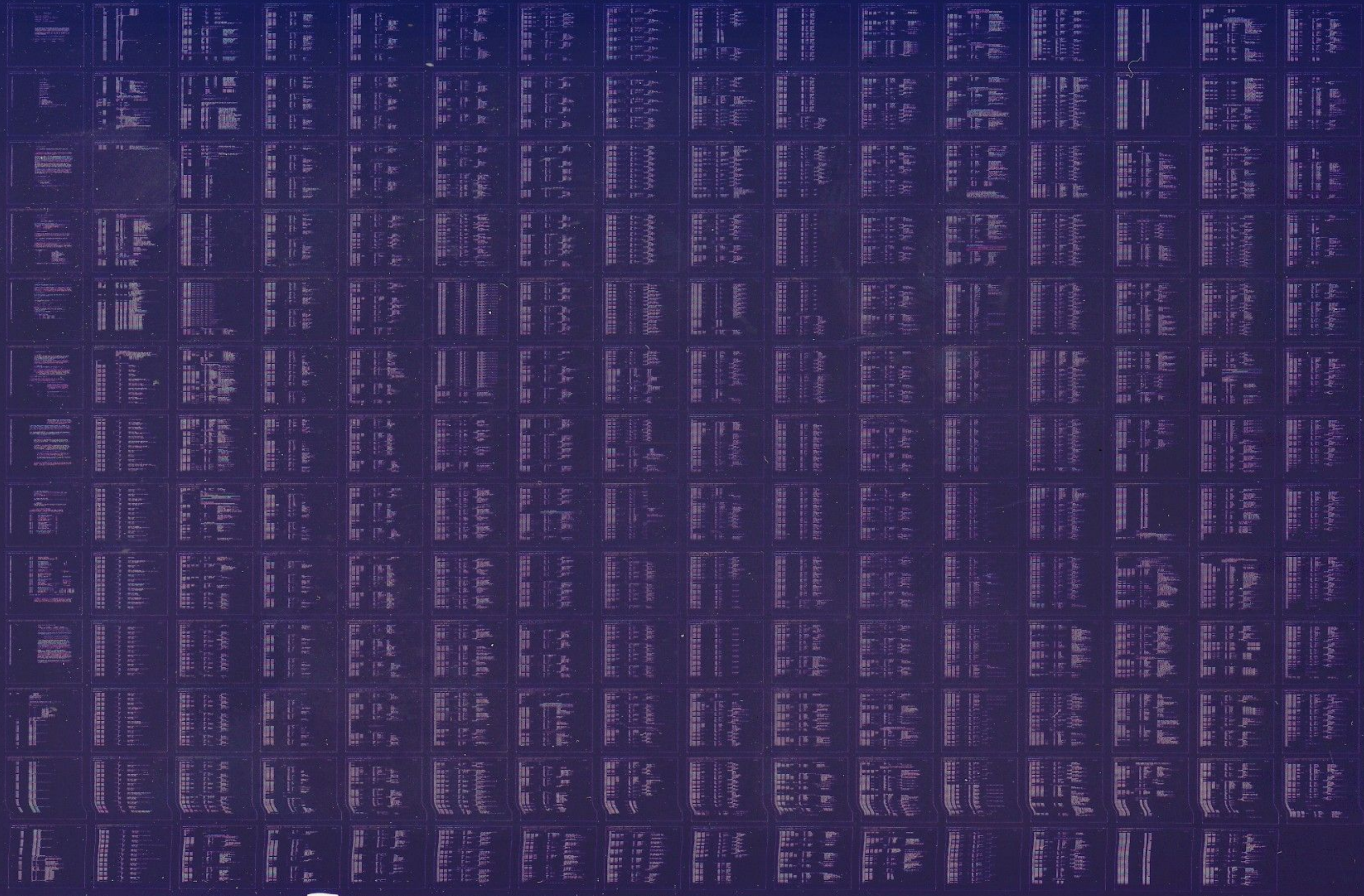


KDJ11-B

KDJ11-B,-B/F CLUSTER
COKDAE0

AH-T854E-MC
1 OF 3 OCT 1985
COPYRIGHT© 1984-85

digital
MADE IN USA



KDJ11-B

KDJ11-B,-B/F CLUSTER
COKDAE0

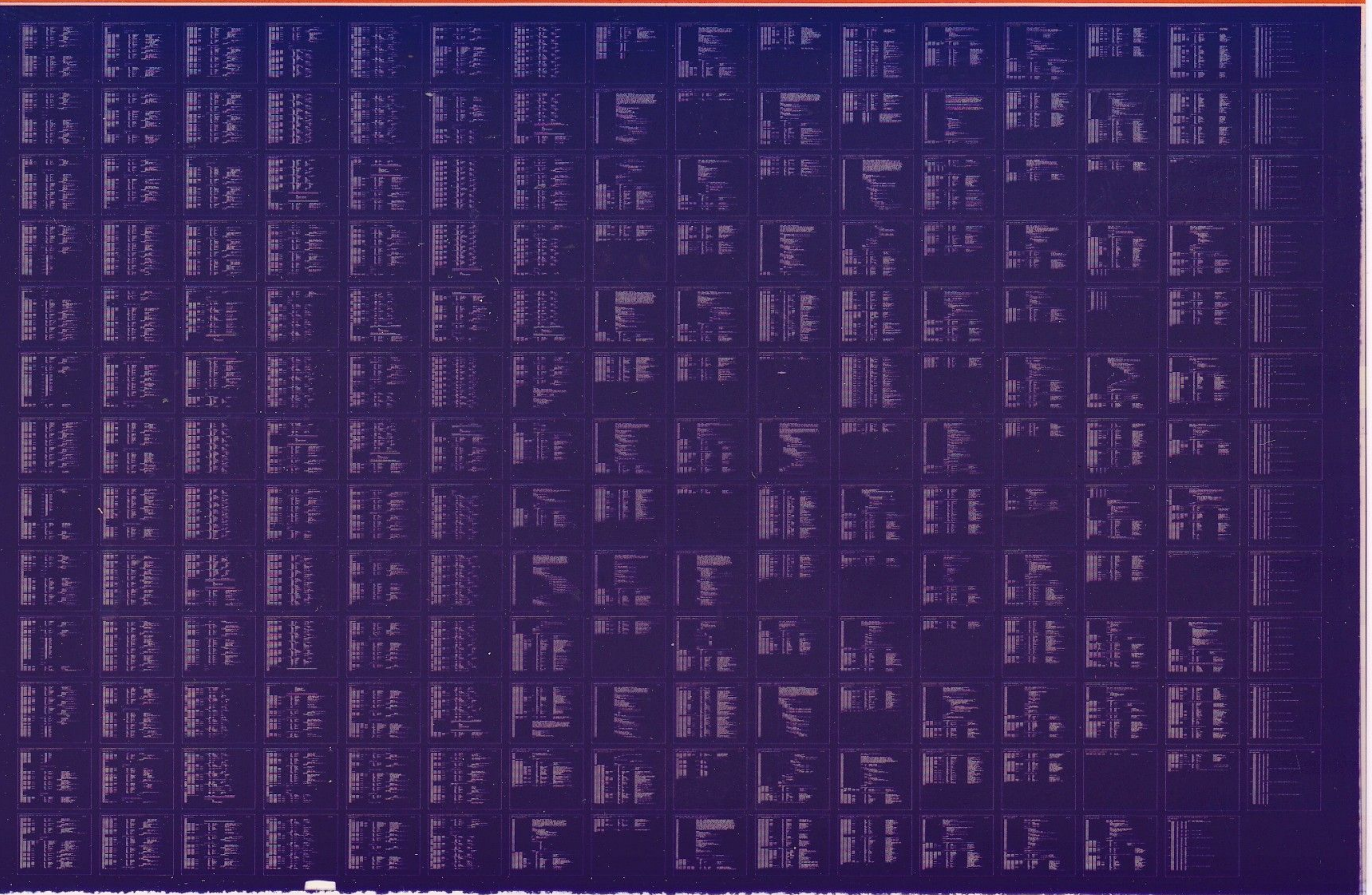
AH-T854E-MC

2 OF 3 OCT 1985

COPYRIGHT© 1984-85

digital

MADE IN USA



KDJ11-B

KDJ11-B,-B/F CLUSTER
COKDAE0

AH-T854E-MC
3 OF 3 OCT 1985
COPYRIGHT© 1984-85

digital
MADE IN USA

The image shows a grid of 36 small, illegible data tables arranged in 12 rows and 3 columns on the left side of the page. Each table appears to be a small data set or report, but the text is too small to read. The tables are organized in a structured layout, with each row containing three tables. The content within these tables is not discernible due to the low resolution and high contrast of the scan.

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

.REM 6

IDENTIFICATION

PRODUCT CODE: AC T853E-MC
PRODUCT NAME: COKDAEO KDJ11-B CLUSTER DIAG.
PRODUCT DATE: MAY, 1985
MAINTAINER: DIAGNOSTIC ENGINEERING

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

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1984,1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

44
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

TABLE OF CONTENTS

- 1. PROGRAM ABSRACT
- 2. SYSTEM REQUIREMENTS
- 3. LOADING AND STARTING PROCEDURES
- 4. SPECIAL ENVIRONMENTS
- 5. PROGRAM OPTIONS
- 6. EXECUTION TIMES
- 7. ERROR INFORMATION
- 8. EXAMPLES
- 9. PROGRAM DESCRIPTION
 - 9.1 J11 CODE
 - 9.2 CACHE CODE
 - 9.3 ON-BOARD ROM CODE
 - 9.4 LINE TIME CLOCK CODE
 - 9.5 SERIAL LINE UNIT CODE
 - 9.6 Q22BE CODE
 - 9.7 LIST of SUBTESTS showing CACHE/APT dependency
- 10. PROGRAM UPDATES AND MODIFICATIONS

79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123

1. PROGRAM ABSTRACT

**** see special instructions for CACHE testing under APT, ****
**** section 9.2, and EEROM testing under APT, section 9.3 ****

This program tests out KDJ11-B CPU board, including the J11 chip set, on-board cache, on-board ROM's, including 16 bit and the 8 bit EEROM, serial line unit, and line time clocks.

The KDJ11-B is a PDP-11 CPU that incorporates the J11 chip set as the heart of the processor. It is a quad height Q22 bus module. It has on-board cache, some of the functionality of the cache is hidden inside the J11 and the rest of the functions implemented in two on-board gate arrays. The storage capacity of the cache is 4K bytes of RAM, called data RAM's. The cache is implemented as a direct mapped cache with address bits 21 through 13 stored in a different set of RAM's called tag store.

The KDJ11-B also has two on-board ROM's. One of them, the 16-bit addressable ROM, contains the self-test and the boot codes. The other ROM, the 8-bit addressable one, contains the base area with hardware selection parameters, optional bootstraps, optional UFD (User Friendly Diagnostic) system description area, and optional foreign language text.

The Serial Line Unit is implemented thru a D1art chip which provides the standard console interface to the CPU. It has internal loop back mode and provides with three clock lines: 800HZ, 60HZ, and 50HZ.

The line time clock functions are implemented using those three lines and the BEVENT line. The line clock status register is implemented in one of the gate arrays.

2. SYSTEM REQUIREMENTS

Hardware Requirements

To run successfully the diagnostic needs:

1. KDJ11-B CPU module
2. console terminal
3. at least 28K of memory

In DVT, and stage one manufacturing (module assembly) the Q22 Bus exerciser is needed to check Q22 Bus logic.

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
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179

3. LOADING AND STARTING PROCEDURES

To start-up this program:

1. Boot XXDP.
2. Type "R NAME", where name is the name of the BIN or BIC file for this program.

The starting address of the program is 200.

Note: if trying to restart the program in an arbitrary place after HALT on Break the following registers should be set up:

1777572=0 to disable memory management
1777520=1000 to clear diagnostic mode (bit 8), but still save HALT on Break
1777746=400 to flush the cache

4. SPECIAL ENVIRONMENTS

The program is APT compatible. It can also be run under the UFD monitor. In those cases none of the standard error printouts occur. Refer to corresponding documents on running procedures in APT and under UFD monitor.

**** see special instructions for CACHE testing under APT, ****
**** section 9.2, and EEROM testing under APT, section 9.3 ****

5. PROGRAM OPTIONS

The Q22 Bus Exerciser is utilized if it is present in the system and the diagnostic is not running in UFD mode. Standard capabilities of looping on test and on error are provided. In order to run the extensive cache data RAM test bit 7 has to be set in the software switch register. To run the extensive cache tag RAM test bit 6 has to be set in the software switch register.

SWITCH REGISTER SELECTION:

BIT NUMBER	USE
15	HALT ON ERROR
14	LOOP ON PRESENT TEST
13	INHIBIT ERROR TYPEOUTS
*** 12	EEROM subtest RUN switch
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<5-0>
7	DO EXTENSIVE DATA RAM TEST
6	DO EXTENSIVE TAG RAM TEST
5-0	Subtest number to loop on (BIT 8)

*** running with this test will change all but the first 109 bytes of EEROM data! be SURE to read instructions in sect. 9.3

181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221

6. EXECUTION TIMES

Without extensive RAM tests, the diagnostic runs in under 15 seconds. With them, it takes about 2 minutes.

In addition, when the EEROM test bit is on, pass 1 takes an additional amount of time of 1 minute for 15MHZ J-11, to about 40 sec. for the 20MHZ. version. These test times are for the 2K EEROM, for 8K parts, multiply by 4. If running under APT, the system manager must be sure the first pass run time allows adequate time for completion of this test!

7. ERROR INFORMATION

In the case of errors, a failing PC and test numbers are given. Where it is possible, expected and received data are given. For an example, see section 8.

8. EXAMPLES

After booting XXDP+ and starting the program, the following will appear on the terminal:

```
* KDJ11-B CPU DIAGNOSTIC - COKDAEO *
```

```
SWR = XXXXXX   NEW =
```

where XXXXXX correspond to present software switch register setting.

After "NEW" an operator can do one of the following:

- 1) type in a new software switch register setting followed by carriage return or
- 2) just type in carriage return in which case the software register will remain unchanged.

Example of error printout:

```
ERROR IN TAG STORE
TEST  ERROR  ADDRESS ADDRESS
*    PC    <21-16> <15-0>
27   105620 66600  000000
```

Note: this may not correspond to the actual Program Counter.

223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272

9. PROGRAM DESCRIPTION

9.1 J11 CODE

This portion of the code tests out the J11 chip set. It is broken into 3 pieces: CPU tests, which verify different instructions in different modes and different trap conditions; MMU tests, which verify different functions of MMU; and FFP tests, which do different floating point instructions. This portion of the code have been written in close relationship with the J11 microcode. Therefore, even though not all possible instructions in all possible addressing mode have been tested, an attempt has been made to exercise all of the microcode.

9.2 CACHE CODE

This portion of the diagnostic verifies all cache functions. Data and tag RAM's tests are also included.

Note: in order to run extensive cache RAM's tests the corresponding bits in the software switch register should be set. See section 5.
----In particular, the number in \$USWR (sware reg #2) should equal the "first pass run time" set up at installation---- (the exception is when \$USWR is set to ZERO, no condition applies)

TESTING CACHE FUNCTIONS UNDER APT

The testing of Cache related functions under APT is facilitated by use of \$USWR containing a number which will indicate the number of passes with inclusive cache tests as follows:

(all numbers in decimal. \$USWR loaded by APT manager)

default setting	\$USWR = 0	one pass with all subtests followed by continuous testing of non-cache dependent subtests until APT Break is encountered (APT system must not BREAK in for this "First Pass Run Time", now assumed to be approx 16 sec.)
	\$USWR < 16	only non-cache dependent tests will be run APT may break anytime
	\$USWR = 16	same as \$USWR = 0
	\$USWR > 16	\$USWR is divided by 16 (pass time) to get a

274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

number of passes which include all subtests (including cache) After this number of passes, continuous testing with only non-cache dependent sub-tests is continued during which time APT may break with no bad effects.

In this case, the APT setup must not attempt to break for a length of time equal to \$USWR contents, in seconds. The nominal 16 seconds per pass, and the actual pass time of about 10 seconds (which changes as tests may be added) allow a safety margin adding additional tests. The actual number of passes done will be equal to \$USWR divided by 16, (if not 0, or < 16).

!!!!!!!!!!!!!!

***** The guarantee that APT will not attempt to break into the diagnostic for a given number of seconds is provided by the APT system manager loading the variable \$PASTM with the same number \$USWR received. If this is not done, the message "hung diagnostic" will likely be received*****

!!!!!!!!!!!!!!

9.3 ON-BOARD ROM AND EEROM CODE

These tests verify the checksums of the 16-bit ROM and the base area of the 8-bit EEROM, and a write and read of 1's and 0's to each location of the EEROM. The EEROM may be affected by multiple writes, (>10,000 cycles) so it is advisable to select this test with care.

The testing of the on-board EEROM is accomplished by the setting of bit 12 in the software switch register, which will see that the EEROM test is run once. (Pass 1) It should be noted that this bit may have another use in the SYSMAC macro \$EOP, but is not so used within this program. The BCSR is also set for Halt on Break on during this test, as a troubleshooting aid in stand-alone modes.

*** WHEN RUNNING THIS TEST UNDER APT ***

After loading and starting the program with EEROM test switch on, APT must not (break) interrupt the test until the time given in section 6, EXECUTION TIMES, has passed.

This test should never be "scripted" in a way which would cause the test to be run multiple times. It should be run instead once at the beginning or end of a script.

!! The contents of the EEROM are lost, except for the 109 (decimal) byte representing the SETUP area, which are restored at the end of the EEROM subtest. If the test is interrupted before these locations are restored, unpredictable results may occur.

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

9.4 LINE TIME CLOCKS CODE

This part of the program verifies the functions of all 4 clock lines: 3 of them from the serial line chip (50HZ, 60HZ, 800HZ) and BEVENT line.

Note: in UFD mode only functions corresponding to Boot Rom selection are checked.

9.5 SERIAL LINE UNIT CODE

These tests verify the functionality of the SLU chip utilizing the maintenance mode of the chip.

9.6 Q22BE CODE

These tests verify interrupt arbitration of the KDJ11-B, DMA protocol, and cache functions related to the DMA activities, including PMG counter.

9.7 LIST OF SUBTESTS PERFORMED

The following list represents the sequential order of subtests performed in COKDME0..... subtests which are subject to the APT qualifications of section 9.2 are indicated by a Cache-APT label.

TEST NO.	TEST NAME/FUNCTION	APT-CACHE sel	NOTES.
test 1	base instruction set tests	
test 2	check (cache) register access	yes	
test 3	CCR register bit test	yes	
test 4	force miss test	yes	
test 5	hit/miss register test part 1	yes	
test 6	hit/miss register test	yes	
test 7	byte allocation test	yes	
test 10	PDR bit 15 (bypass) test	yes	
test 11	flush cache test	yes	
test 12	unconditional bypass test	yes	
test 13	write wrong data parity test	yes	
test 14	write wrong tag parity	yes	
test 15	parity abort test	yes	
test 16	parity interrupt test	yes	
test 17	miscellaneous parity test	yes	
test 20	memory system error register test	yes	
test 21	check parity aborts blocked	yes	
test 22	multi-processing instruction	yes	
test 23	data store ram, tests	yes	

376	test 24	tag store ram tests	yes	
377	test 25	standalone mode tests	yes	
378	test 26	moving inversions test data RAr,	yes	
379	test 27	moving inversions for tag store	yes	
380				
381	test 30	PCR read/write bits	
382	test 31	bcsr read/write bits	
383	test 32	16 bit ROM checksum	HOB on
384	test 33	8 bit EEROM checksum test	HOB on,
385	test 34	EEROM checkerboard mem test	note 1.0
386	test 35	LKS bit 7	
387	test 36	LKS interrupt priority	
388	test 37	line clock disable	
389				
390	test 40	unconditional clock interrupts	
391	test 41	resetting LKS	only done pass 1
392	test 42	line clock interrupts	yes	not CACHE dep, APT break sensitiv
e				
393	test 43	maintenance register test	
394	test 44	serial line unit registers	
395	test 45	XCSR bit 7	
396	test 46	RCSR bit 7 and XCSR bit 7	done once in APT
397	test 47	RESET and XCSR<2!0>	first pass only
398				
399	test 50	RESET and interrupt enable	
400	test 51	interrupt priority for SLU	done once in APT
401	test 52	Break condition	done only once
402	test 53	overrun condition	done once in APT
403	test 54	LED'S on test	HOB on
404	test 55	MEMORY mapping	not done chain mode, APT, other than pass 1	
405	test 56	wrong parity abort test	
406	test 57	DMA TAG PARITY in standalone	yes	HOB disabled
407				
408	test 60	DMA tag parity w/o standalone	yes	
409	test 61	DMA write hit cycles	yes	
410	test 62	different levels of interrupts		not UFD, only if Q22BE found
411	test 63	Arbitration bet PIRQ interrupt		not UFD, only if Q22BE found
412	test 64	power down test		not UFD, only if Q22BE found
413	test 65	arbitration between interrupt levels		not UFD, only if Q22BE found
414	test 66	PMG counter	yes	not UFD, only if two Q22BE found

as of july 1984, this is the list of subtests, any added subtests will require editing of this list....

note 1. This test will only be run in first pass, and then only if bit 12 is set in the SWR.... Running this test will DESTROY everything in the EEROM except the first 109 bytes..... UFD area, secondary boots, and local language area's would all need to be restored after running this test. The first pass run time must be adjusted accordingly when running under

424

426
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

10. PROGRAM UPDATES AND MODIFICATIONS

Version COKDADO 2-22 85 Howard L. Marshall

1. Changes made by Howard L. Marshall - denoted by "###"
in comment lines of added or changed code, as of 2 22 85

Added subroutine "DETFPA" (determine floating point accelerator)

This subroutine is called just prior to the error macro for all floating point errors. This subroutine prints 1 of 2 possible error messages. they are:

- A.) ERROR DETECTED IN FLOATING POINT ACCELERATOR CHIP.
- B.) ERROR DETECTED IN J11 FLOATING POINT PROCESSOR.

Of the 2 above error messages, the error message that is printed is determined by the state of the "FPA" flag, bit 8, in the maintenance register. this bit is set to a 1 if the floating point accelerator chip is installed in the cpu board. Otherwise, the bit is cleared which indicates that the floating point accelertor chip is not installed. Based solely on this determination, it is logical to assume that most, if not all floating point errors can be attributed to the FPA chip if the "FPA" bit is set or to the floating point processor within the j11 if the "FPA" bit is cleared.

Version COKDAEO 13-MAY-85 Howard L. Marshall

Changes made by Howard L. Marshall - denoted by "###"

Corrected error in 22-Bit MMU Tests (test #1) which used number 157776 as virtual address intended to address last word of RAM just below the I/O page. This virtual address in conjunction with a value of 177500 in KPAR6 actually maps to physical address 17767776 which is at 2046K (half-way into the I/O page). Changed that virtual address to 147776 which when used with the above value in KPAR6, maps to physical address 17757776 which is in the 2044K page, the last non I/O page address.

Modified test #53, Overrun condition test to allow this test to work properly when the console terminal is running at the minimum ORION baud rate, 300 baud. Changed wait argument from: #150000 to #175000.

E

```

482      167400      $SWR=167400
483      000300      $SWRMK=300
484                                     .TITLE COKDAEO KDJ11-B CLUSTER DIAG.
; *COPYRIGHT (C) MAY 85
; *DIGITAL EQUIPMENT CORP.
; *MAYNARD, MASS. 01754
; *
; *PROGRAM BY DIAG. ENG.
; *
; *THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
; *PACKAGE (MAINDEC-11-DZQAC-C8), OCT, 1982.
; *
485      000001      $TN=1
; *
; *SBTTL OPERATIONAL SWITCH SETTINGS
; *
; *      SWITCH          USE
; *      -----          -
; *      15             HALT ON ERROR
; *      14             LOOP ON TEST
; *      13             INHIBIT ERROR TYPEOUTS
; *      11             INHIBIT ITERATIONS
; *      10             BELL ON ERROR
; *      9              LOOP ON ERROR
; *      8              LOOP ON TEST IN SWR<5:0>
487      ; *      7              DO EXTENSIVE DATA RAM TEST
488      ; *      6              DO EXTENSIVE TAG RAM TEST
489                                     .SBTTL MEMORY MANAGEMENT DEFINITIONS
; *KT11 VECTOR ADDRESS
MMVEC= 250
; *KT11 STATUS REGISTER ADDRESSES
177572      SR0= 177572
177574      SR1= 177574
177576      SR2= 177576
172516      SR3= 172516
; *USER "I" PAGE DESCRIPTOR REGISTERS
177600      UIPDR0= 177600
177602      UIPDR1= 177602
177604      UIPDR2= 177604
177606      UIPDR3= 177606
177610      UIPDR4= 177610
177612      UIPDR5= 177612
177614      UIPDR6= 177614
177616      UIPDR7= 177616
; *USER "D" PAGE DESCRIPTOR REGISTORS
177620      UDPDR0= 177620
177622      UDPDR1= 177622
177624      UDPDR2= 177624
177626      UDPDR3= 177626
177630      UDPDR4= 177630
177632      UDPDR5= 177632
177634      UDPDR6= 177634
177636      UDPDR7= 177636
; *USER "I" PAGE ADDRESS REGISTERS
177640      UIPAR0= 177640
177642      UIPAR1= 177642
177644      UIPAR2= 177644
177646      UIPAR3= 177646

```


MEMORY MANAGEMENT DEFINITIONS

177650	UIPAR4= 177650
177652	UIPAR5= 177652
177654	UIPAR6= 177654
177656	UIPAR7= 177656
	;*USER "D" PAGE ADDRESS REGISTERS
177660	UDPAR0= 177660
177662	UDPAR1= 177662
177664	UDPAR2= 177664
177666	UDPAR3= 177666
177670	UDPAR4= 177670
177672	UDPAR5= 177672
177674	UDPAR6= 177674
177676	UDPAR7= 177676
	;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
172200	SIPDR0= 172200
172202	SIPDR1= 172202
172204	SIPDR2= 172204
172206	SIPDR3= 172206
172210	SIPDR4= 172210
172212	SIPDR5= 172212
172214	SIPDR6= 172214
172216	SIPDR7= 172216
	;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
172220	SDPDR0= 172220
172222	SDPDR1= 172222
172224	SDPDR2= 172224
172226	SDPDR3= 172226
172230	SDPDR4= 172230
172232	SDPDR5= 172232
172234	SDPDR6= 172234
172236	SDPDR7= 172236
	;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
172240	SIPAR0= 172240
172242	SIPAR1= 172242
172244	SIPAR2= 172244
172246	SIPAR3= 172246
172250	SIPAR4= 172250
172252	SIPAR5= 172252
172254	SIPAR6= 172254
172256	SIPAR7= 172256
	;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
172260	SDPAR0= 172260
172262	SDPAR1= 172262
172264	SDPAR2= 172264
172266	SDPAR3= 172266
172270	SDPAR4= 172270
172272	SDPAR5= 172272
172274	SDPAR6= 172274
172276	SDPAR7= 172276
	;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
172300	KIPDR0= 172300
172302	KIPDR1= 172302
172304	KIPDR2= 172304
172306	KIPDR3= 172306
172310	KIPDR4= 172310
172312	KIPDR5= 172312
172314	KIPDR6= 172314

MEMORY MANAGEMENT DEFINITIONS

490

```

172316      KIPDR7= 172316
            ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
172320      KDPDR0= 172320
172322      KDPDR1= 172322
172324      KDPDR2= 172324
172326      KDPDR3= 172326
172330      KDPDR4= 172330
172332      KDPDR5= 172332
172334      KDPDR6= 172334
172336      KDPDR7= 172336
            ;*KERNEL "I" PAGE ADDRESS REGISTERS
172340      KIPAR0= 172340
172342      KIPAR1= 172342
172344      KIPAR2= 172344
172346      KIPAR3= 172346
172350      KIPAR4= 172350
172352      KIPAR5= 172352
172354      KIPAR6= 172354
172356      KIPAR7= 172356
            ;*KERNEL "D" PAGE ADDRESS REGISTERS
172360      KDPAR0= 172360
172362      KDPAR1= 172362
172364      KDPAR2= 172364
172366      KDPAR3= 172366
172370      KDPAR4= 172370
172372      KDPAR5= 172372
172374      KDPAR6= 172374
172376      KDPAR7= 172376
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100      STACK= 1100
104000      ERROR= EMT                ;;BASIC DEFINITION OF ERROR CALL
000004      SCOPE= IOT                ;;BASIC DEFINITION OF SCOPE CALL
;*MISCELLANEOUS DEFINITIONS
000011      HT= 11                    ;;CODE FOR HORIZONTAL TAB
000012      LF= 12                    ;;CODE FOR LINE FEED
000015      CR= 15                    ;;CODE FOR CARRIAGE RETURN
000200      CRLF= 200                 ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776      PS= 177776                ;;PROCESSOR STATUS WORD
177776      PSW= PS
177774      STKLMT= 177774            ;;STACK LIMIT REGISTER
177772      PIRQ= 177772              ;;PROGRAM INTERRUPT REQUEST REGISTER
177570      DSWR= 177570              ;;HARDWARE SWITCH REGISTER
177570      DDISP= 177570            ;;HARDWARE DISPLAY REGISTER
;*GENERAL PURPOSE REGISTER DEFINITIONS
000000      R0= #0                    ;;GENERAL REGISTER
000001      R1= #1                    ;;GENERAL REGISTER
000002      R2= #2                    ;;GENERAL REGISTER
000003      R3= #3                    ;;GENERAL REGISTER
000004      R4= #4                    ;;GENERAL REGISTER
000005      R5= #5                    ;;GENERAL REGISTER
000006      R6= #6                    ;;GENERAL REGISTER
000007      R7= #7                    ;;GENERAL REGISTER
000006      SP= #6                    ;;STACK POINTER
000007      PC= #7                    ;;PROGRAM COUNTER
;*PRIORITY LEVEL DEFINITIONS
000000      PRO= 0                    ;;PRIORITY LEVEL 0

```


BASIC DEFINITIONS

```

0000- PR1= 40 ;;PRIORITY LEVEL 1
000 PR2= 100 ;;PRIORITY LEVEL 2
000 PR3= 110 ;;PRIORITY LEVEL 3
000200 PR4= 200 ;;PRIORITY LEVEL 4
000240 PR5= 240 ;;PRIORITY LEVEL 5
000300 PR6= 300 ;;PRIORITY LEVEL 6
000340 PR7= 340 ;;PRIORITY LEVEL 7
;*"SWITCH REGISTER' SWITCH DEFINITIONS
100000 SW15= 100000
040000 SW14= 40000
020000 SW13= 20000
010000 SW12= 10000
004000 SW11= 4000
002000 SW10= 2000
001000 SW09= 1000
000400 SW08= 400
000200 SW07= 200
000100 SW06= 100
000040 SW05= 40
000020 SW04= 20
000010 SW03= 10
000004 SW02= 4
000002 SW01= 2
000001 SW00= 1
001000 SW9= SW09
000400 SW8= SW08
000200 SW7= SW07
000100 SW6= SW06
000040 SW5= SW05
000020 SW4= SW04
000010 SW3= SW03
000004 SW2= SW02
000002 SW1= SW01
000001 SW0= SW00
;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BI.03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
001000 BIT9= BIT09
000400 BIT8= BIT08
000200 BIT7= BIT07
000100 BIT6= BIT06
000040 BIT5= BIT05
000020 BIT4= BIT04

```

BASIC DEFINITIONS

```

000010      BIT3= BIT03
000004      BIT2= BIT02
000002      BIT1= BIT01
000001      BIT0= BIT00
; *BASIC "CPU" TRAP VECTOR ADDRESSES
000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
000010      RESVEC= 10       ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014      TBITVEC=14      ;; "T" BIT
000014      TRTVEC= 14      ;; TRACE TRAP
000014      BPTVEC= 14      ;; BREAKPOINT TRAP (BPT)
000020      IOTVEC= 20      ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024      PWRVEC= 24      ;; POWER FAIL
000030      EMTVEC= 30      ;; EMULATOR TRAP (EMT) **ERROR**
000034      TRAPVEC=34     ;; "TRAP" TRAP
000060      TKVEC= 60       ;; TTY KEYBOARD VECTOR
000064      TPVEC= 64       ;; TTY PRINTER VECTOR
000240      PIRQVEC=240    ;; PROGRAM INTERRUPT REQUEST VECTOR
491         UFDSET= 1      ;; FLAG FOR UFD
492         .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
000174      000000      DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
000176      000000      SWREG: .WORD 0           ;; SOFTWARE SWITCH REGISTER
493         000200      . = 200
494 000200      005037      001160      CLR      $TMP0
495 000204      000137      004024      JMP      @START
496         000220      . = 220
497 000220      012737      000777      001160      MOV      @777, $TMP0
498 000226      000137      004024      JMP      @START
^99
.00         .SBTTL ACT11 HOOKS
; *****
; HOOKS REQUIRED BY ACT11
           $SVPC= .          ;; SAVE PC
           . = 46
000046      140440      $ENDAD          ;; 1) SET LOC. 46 TO ADDRESS OF $ENDAD IN .SEOP
           . = 52
000052      000000      .WORD 0        ;; 2) SET LOC. 52 TO ZERO
           . = $SVPC        ;; RESTORE PC
501         .SBTTL APT PARAMETER BLOCK
; *****
; SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
; *****
           . $X= .          ;; SAVE CURRENT LOCATION
           . = 24          ;; SET POWER FAIL TO POINT TO START OF PROGRAM
000024      000200      200          ;; FOR APT START UP
           . = 44          ;; POINT TO APT INDIRECT ADDRESS PNTR.
000044      000232      $APTHDR      ;; POINT TO APT HEADER BLOCK
           . = . $X        ;; RESET LOCATION COUNTER
; *****
; SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
; INTERFACE SPEC.
000232      000000      $APTHD:
000232      000000      $HIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

```


APT PARAMETER BLOCK

000234	001200	\$MBADR: .WORD	\$MAIL	:: ADDRESS OF APT MAILBOX (BITS 0 15)
000236	000000	\$TSTM: .WORD		:: RUN TIM OF LONGEST TEST
000240	000300	\$PASTM: .WORD		:: RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000242	000000	\$UNITM: .WORD		:: ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000244	000052	.WORD	\$E*END-\$MAIL/2	:: LENGTH MAILBOX-ETABLE(WORDS)

COMMON TAGS

502

```

.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
      .-1100
001100 001100 $CMTAG:                ;; START OF COMMON TAGS
001100 000000          .WORD 0
001102 000          $TSTNM: .BYTE 0                ;; CONTAINS THE TEST NUMBER
001103 000          $ERFLG: .BYTE 0                ;; CONTAINS ERROR FLAG
001104 000000        $ICNT:  .WORD 0                ;; CONTAINS SUBTEST ITERATION COUNT
001106 000000        $LPADR: .WORD 0                ;; CONTAINS SCOPE LOOP ADDRESS
001110 000000        $LPERR: .WORD 0                ;; CONTAINS SCOPE RETURN FOR ERRORS
001112 000000        $ERTTL: .WORD 0                ;; CONTAINS TOTAL ERRORS DETECTED
001114 000          $ITEMB: .BYTE 0                ;; CONTAINS ITEM CONTROL BYTE
001115 001          $ERMAX: .BYTE 1                ;; CONTAINS MAX. ERRORS PER TEST
001116 000000        $ERRPC: .WORD 0                ;; CONTAINS PC OF LAST ERROR INSTRUCTION
001120 000000        $GDADR: .WORD 0                ;; CONTAINS ADDRESS OF 'GOOD' DATA
001122 000000        $BDADR: .WORD 0                ;; CONTAINS ADDRESS OF 'BAD' DATA
001124 000000        $GDDAT: .WORD 0                ;; CONTAINS 'GOOD' DATA
001126 000000        $BDDAT: .WORD 0                ;; CONTAINS 'BAD' DATA
001130 000000          .WORD 0                ;; RESERVED--NOT TO BE USED
001132 000000          .WORD 0
001134 000          $AUTOB: .BYTE 0                ;; AUTOMATIC MODE INDICATOR
001135 000          $INTAG: .BYTE 0                ;; INTERRUPT MODE INDICATOR
001136 000000          .WORD 0
001140 177570        SWR:   .WORD DSWR                ;; ADDRESS OF SWITCH REGISTER
001142 177570        DISPLAY: .WORD DDISP                ;; ADDRESS OF DISPLAY REGISTER
001144 177560        $TKS:   177560                ;; TTY KBD STATUS
001146 177562        $TKB:   177562                ;; TTY KBD BUFFER
001150 177564        $TPS:   177564                ;; TTY PRINTER STATUS REG. ADDRESS
001152 177566        $TPB:   177566                ;; TTY PRINTER BUFFER REG. ADDRESS
001154 000          $NULL:  .BYTE 0                ;; CONTAINS NULL CHARACTER FOR FILLS
001155 002          $FILLS: .BYTE 2                ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
001156 012          $FILLC: .BYTE 12               ;; INSERT FILL CHARS. AFTER A "LINE FEED"
001157 000          $TPFLG: .BYTE 0                ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
      .REPT 2
001160 000000        $TMPO:  .WORD 0                ;; USER DEFINED
001162 000000        $TMP1:  .WORD 0                ;; USER DEFINED
001164 000000        $TIMES: 0                    ;; MAX. NUMBER OF ITERATIONS
001166 000000        $ESCAPE:0                    ;; ESCAPE ON ERROR ADDRESS
001170 207          377 377 $BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
001173 000
001174 077          $QUES:  .ASCII /?/                ;; QUESTION MARK
001175 015          $CRLF:  .ASCII <15>                ;; CARRIAGE RETURN
001176 012          000 $LF:   .ASCIZ <12>                ;; LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
001200 $MAIL:                ;; APT MAILBOX
001200 000070 $MSGTY: .WORD AMSTY ;; MESSAGE TYPE CODE
001202 000000 $FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER
001204 000000 $TESTN: .WORD ATESTN ;; TEST NUMBER
001206 000000 $PASS:  .WORD APASS  ;; PASS COUNT
001210 000000 $DEVCT: .WORD ADEVCT ;; DEVICE COUNT
001212 000000 $UNIT:  .WORD AUNIT  ;; I/O UNIT NUMBER
001214 000000 $MSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS

```


APT MAILBOX-ETABLE

```

001216 000000 $MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
001220 $ETABLE: ;;APT ENVIRONMENT TABLE
C^1220 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
001221 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
001222 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
001224 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
0C1226 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
;*
;* BITS 15-11=CPU TYPE
;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
;* 11/70=06,PDQ=07,Q=10
;* BIT 10=REAL TIME CLOCK
;* BIT 9=FLOATING POINT PROCESSOR
;* BIT 8=MEMORY MANAGEMENT
001230 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
001231 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
;*
;* MEM.TYPE BYTE -- (HIGH BYTE)
;* 900 NSEC CORE=001
;* 300 NSEC BIPOLAR=002
;* 500 NSEC MOS=003
001232 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
;*
;* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
001235 000 $MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2
0C1236 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
001240 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
001241 000 $MTYP3: .BYTE AMTYP3 ;;MEM. TYPE,BLK#3
001242 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
001244 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
001245 000 $MTYP4: .BYTE AMTYP4 ;;MEM. TYPE,BLK#4
001246 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
001250 000000 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001252 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001254 000000 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001256 000000 $DEVM: .WORD ADEVM ;;DEVICE MAP
0C1260 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
001262 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
001264 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTOR WORD#0
001266 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTOR WORD#1
001270 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTOR WORD#2
001272 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTOR WORD#3
001274 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTOR WORD#4
001276 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTOR WORD#5
001300 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTOR WORD#6
001302 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTOR WORD#7
001304 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTOR WORD#8
001306 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTOR WORD#9
001310 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
001312 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
001314 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
001316 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
001320 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
001322 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
001324 $ETEND:

```

ERROR POINTER TABLE

```
.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 $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
;*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
$ERRTB:
```

001324

503

504

505

506

507 001324 127002

508 001326 134621

509 001330 136026

510 001332 000000

511

512 001334 127036

513 001336 134621

514 001340 136026

515 001342 000000

516

517 001344 127050

518 001346 134621

519 001350 136026

520 001352 000000

521

522 001354 127062

523 001356 134646

524 001360 136034

525 001362 000000

526

527 001364 127122

528 001366 134733

529 001370 136046

530 001372 000000

531

532 001374 127155

533 001376 134733

534 001400 136046

535 001402 000000

536

537 001404 127220

538 001406 135032

539 001410 136062

540 001412 000000

541

542 001414 127254

543 001416 135032

544 001420 136062

545 001422 000000

546

547 001424 127275

548 001426 135032

.SBTTL ERROR DEFINITIONS

;ITEM 1

```
EM1          ;CPU ERROR
DH1          ;TEST #, ERROR PC
DT1          ;$TMP1,$ERRPC
0
```

;ITEM 2

```
EM2          ;MMU ERROR
DH1          ;TEST #, ERROR PC
DT1          ;$TMP1,$ERRPC
0
```

;ITEM 3

```
EM3          ;FPP ERROR
DH1          ;TEST #, ERROR PC
DT1          ;$TMP1,$ERRPC
0
```

;ITEM 4

```
EM4          ;ERROR IN READ-WRITE BITS OF CCR
DH4          ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
DT4          ;$TMP1,$ERRPC,R1,CCR
0
```

;ITEM 5

```
EM5          ;FORCE MISS WRITES TO CACHE
DH5          ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
DT5          ;$TMP1,$ERRPC, R2, R1, $GDDAT
0
```

;ITEM 6

```
EM6          ;FORCE MISS WRITE INVALIDATES CACHE
DH5          ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
DT5          ;$TMP1,$ERRPC, R2, R1, $GDDAT
0
```

;ITEM 7

```
EM7          ;UNEXPECTED PARITY INTERRUPT
DH7          ;TEST #, PC, ADDRESS ACCESSED, MSER
DT7          ;$TMP1,$ERRPC,$BDADR,MSER
0
```

;ITEM 10

```
EM10         ;TAG PARITY ERROR
DH7          ;TEST #, PC, ADDRESS ACCESSED, MSER
DT7          ;$TMP1,$ERRPC,$BDADR,MSER
0
```

;ITEM 11

```
EM11         ;DATA PARITY ERROR
DH7          ;TEST #, PC, ADDRESS ACCESSED, MSER
```


ERROR DEFINITIONS

549	001430	136062	DT7	;	TMP1,ERRPC,BDADR,MSER
550	001432	000000	0		
551			;	ITEM 12	
552	001434	127317	EM12	;	LOW BYTE PARITY ERROR
553	001436	135032	DH7	;	TEST #, PC, ADDRESS ACCESSED, MSER
554	001440	136062	DT7	;	TMP1,ERRPC,BDADR,MSER
555	001442	000000	0		
556			;	ITEM 13	
557	001444	127345	EM13	;	HIGH BYTE PARITY ERROR
558	001446	135032	DH7	;	TEST #, PC, ADDRESS ACCESSED, MSER
559	001450	136062	DT7	;	TMP1,ERRPC,BDADR,MSER
560	001452	000000	0		
561			;	ITEM 14	
562	001454	127374	EM14	;	ERROR DATA PATH
563	001456	134646	DH4	;	TEST #, PC, EXPECTED DATA, RECEIVED DATA
564	001460	136074	DT14	;	TMP1,ERRPC,GDDAT,TSTLOC
565	001462	000000	0		
566			;	ITEM 15	
567	001464	127417	EM15	;	FORCE MISS READS FROM CACHE
568	001466	134733	DH5	;	TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
569	001470	136046	DT5	;	TMP1,ERRPC,R2,R1,GDDAT
570	001472	000000	0		
571			;	ITEM 16	
572	001474	127453	EM16	;	FORCE MISS READS FROM CACHE AND MISS
573	001476	134733	DH5	;	TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
574	001500	136046	DT5	;	TMP1,ERRPC,R2,R1,GDDAT
575	001502	000000	0		
576			;	ITEM 17	
577	001504	127520	EM17	;	ERROR IN RECORDING HITS IN HIT/MISS
578	001506	134646	DH4	;	TEST #, PC, EXPECTED DATA, RECEIVED DATA
579	001510	136106	DT17	;	TMP1,ERRPC,GDDAT,RECDAT
580	001512	000000	0		
581			;	ITEM 20	
582	001514	127564	EM20	;	WRITE BYTE ALLOCATES CACHE
583	001516	134621	DH1	;	TEST #, PC
584	001520	136026	DT1	;	TMP1,ERRPC
585	001522	000000	0		
586			;	ITEM 21	
587	001524	127617	EM21	;	WRITE BYTE HIT DOES NOT RECORD HIT
588	001526	134621	DH1	;	TEST #, PC
589	001530	136026	DT1	;	TMP1,ERRPC
590	001532	000000	0		
591			;	ITEM 22	
592	001534	127662	EM22	;	BYTES REVERSED ON WRITE CYCLES
593	001536	134621	DH1	;	TEST #, PC
594	001540	136026	DT1	;	TMP1,ERRPC
595	001542	000000	0		
596			;	ITEM 23	
597	001544	127721	EM23	;	CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
598	001546	134733	DH5	;	TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
599	001550	136046	DT5	;	TMP1,ERRPC,R2,R1,GDDAT
600	001552	000000	0		
601			;	ITEM 24	
602	001554	127775	EM24	;	HITS RECORDED AFTER FLUSHING CACHE
603	001556	135105	DH24	;	TEST #, PC, NUMBER OF HITS
604	001560	136120	DT24	;	TMP1,ERRPC,R3
605	001562	000000	0		

ERROR DEFINITIONS

606				
607	001564	130040	:ITEM 25	EM25 ;BYPASS DOESN'T INVALIDATE CACHE
608	001566	134621		DH1 ;TEST #, PC
609	001570	136026		DT1 ;\$TMP1,\$ERRPC
610	001572	000000		0
611			:ITEM 26	EM26 ;MSER DOES NOT CLEAR ON WRITE REFERENCE
612	001574	130100		DH1 ;TEST #, PC
613	001576	134621		DT1 ;\$TMP1,\$ERRPC
614	001600	136026		0
615	001602	000000		
616			:ITEM 27	EM27 ;PARITY ERROR DON'T CAUSE A MISS
617	001604	130147		DH27 ;TEST #, PC, MSER, HIT/MISS
618	001606	135151		DT27 ;\$TMP1,\$ERRPC,MSER,R3,0
619	001610	136130		0
620	001612	000000		
621			:ITEM 30	EM30 ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
622	001614	130207		DH27 ;TEST #, PC, MSER, HIT/MISS
623	001616	135151		DT27 ;\$TMP1,\$ERRPC,MSER,R3,0
624	001620	136130		0
625	001622	000000		
626			:ITEM 31	EM31 ;PARITY ERROR IGNORED
627	001624	130261		DH1 ;TEST #, PC
628	001626	134621		DT1 ;\$TMP1,\$ERRPC
629	001630	136026		0
630	001632	000000		
631			:ITEM 32	EM32 ;PARITY ERROR IGNORED ON LOW BYTE
632	001634	130306		DH1 ;TEST #, PC
633	001636	134621		DT1 ;\$TMP1,\$ERRPC
634	001640	136026		0
635	001642	000000		
636			:ITEM 33	EM33 ;PARITY ERROR IGNORED ON HIGH BYTE
637	001644	130347		DH1 ;TEST #, PC
638	001646	134621		DT1 ;\$TMP1,\$ERRPC
639	001650	136026		0
640	001652	000000		
641			:ITEM 34	EM34 ;PARITY ABORT LOGIC DOESN'T WORK
642	001654	130411		DH1 ;TEST #, PC
643	001656	134621		DT1 ;\$TMP1,\$ERRPC
644	001660	136026		0
645	001662	000000		
646			:ITEM 35	EM35 ;MSER NOT SET PROPERLY
647	001664	130451		DH4 ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
648	001666	134646		DT35 ;\$TMP1,\$ERRPC,\$GDDAT,MSER,0
649	001670	136142		0
650	001672	000000		
651			:ITEM 36	EM36 ;PARITY INTERRUPT DOESN'T WORK
652	001674	130477		DH1 ;TEST #, PC
653	001676	134621		DT1 ;\$TMP1,\$ERRPC
654	001700	136026		0
655	001702	000000		
656			:ITEM 37	EM37 ;NXM AND PARITY ABORT DIN'T HAPPEN
657	001704	130543		DH1 ;TEST #, PC
658	001706	134621		DT1 ;\$TMP1,\$ERRPC
659	001710	136026		0
660	001712	000000		
661			:ITEM 40	EM40 ;PARITY ABORT NOT BLOCKED BY NXM TRAP
662	001714	130605		

ERROR DEFINITIONS

663	001716	134621	DH1	:TEST #, PC
664	001720	136026	DT1	;\$TMP1,\$ERRPC
665	001722	000000	0	
666			:ITEM 41	
667	001724	130652	EM41	:MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS
668	001726	135214	DH41	:TEST #, PC, INSTRUCTION OPCODE
669	001730	136154	DT41	;\$TMP1,\$ERRPC,\$BDDAT
670	001732	000000	0	
671			:ITEM 42	
672	001734	130736	EM42	:ERROR IN PARITY LOGIC
673	001736	134621	DH1	:TEST #, PC
674	001740	136026	DT1	;\$TMP1,\$ERRPC
675	001742	000000	0	
676			:ITEM 43	
677	001744	130764	EM43	:ERROR IN CACHE DATA RAMS
678	001746	135265	DH43	:TEST #, PC, EXPECTED DATA, RECEIVED DATA, CACHE LOCATION
679	001750	136164	DT43	;\$TMP1,\$ERRPC,R1,RECDAT,\$BDAOR
680	001752	000000	0	
681			:ITEM 44	
682	001754	131015	EM44	:ERROR IN NXM IN STANDALONE MODE
683	001756	134621	DH1	:TEST #, PC
684	001760	136026	DT1	;\$TMP1,\$ERRPC
685	001762	000000	0	
686			:ITEM 45	
687	001764	131055	EM45	:HITS NOT RECORDED PROPERLY THRU HIT/MISS
688	001766	134621	DH1	:TEST #, PC
689	001770	136026	DT1	;\$TMP1,\$ERRPC
690	001772	000000	0	
691			:ITEM 46	
692	001774	131137	EM46	:HIT RECORDED FOR A LOCATION NOT IN CACHE
693	001776	135365	DH47	:TEST #, PC, LOCATION ACCESSED
694	002000	136200	DT47	;\$TMP1,\$ERRPC,KIPAR6,\$BDADR
695	002002	000000	0	
696			:ITEM 47	
697	002004	131227	EM47	:MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE
698	002006	135365	DH47	:TEST #, PC, LOCATION ACCESSED
699	002010	136200	DT47	;\$TMP1,\$ERRPC,KIPAR6,\$BDADR
700	002012	000000	0	
701			:ITEM 50	
702	002014	131314	EM50	:ERROR IN TAG STORE
703	002016	135365	DH47	:TEST #, PC, ADDRESS
704	002020	136212	DT50	;\$TMP1,\$ERRPC,R1,\$BDADR
705	002022	000000	0	
706			:ITEM 51	
707	002024	131337	EM51	:ERROR PCR READ WRITE BITS
708	002026	134646	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
709	002030	136224	DT51	;\$TMP1,\$ERRPC,\$GDDAT,PCR
710	002032	000000	0	
711			:ITEM 52	
712	002034	131371	EM52	:ERROR IN BCSR READ-WRITE BITS
713	002036	134646	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
714	002040	136236	DT52	;\$TMP1,\$ERRPC,\$GDDAT,BCSR
715	002042	000000	0	
716			:ITEM 53	
717	002044	131427	EM53	:RESET DOESN'T CLEAR BCSR<4>
718	002046	134621	DH1	:TEST #, PC
719	002050	136026	DT1	;\$TMP1,\$ERRPC

ERROR DEFINITIONS

```

720 002052 000000      0
721                ;ITEM 54
722 002054 131463      EM54      ;CHECKSUM ERROR IN 16-BIT ROM
723 002056 134621      DH1        ;TEST @, PC
724 002060 136026      DT1        ;$TMP1,$ERRPC
725 002062 000000      0
726                ;ITEM 55
727 002064 131521      EM55      ;CHCKSUM ERPOR IN 8-BIT ROM
728 002066 134621      DH1        ;TEST @, PC
729 002070 136026      DT1        ;$TMP1,$ERRPC
730 002072 000000      0
731                ;JTEM 56
732 002074 131555      EM56      ;TIMEOUT READING LKS
733 002076 134621      DH1        ;TEST @, PC
734 002100 136026      DT1        ;$TMP1,$ERRPC
735 002102 000000      0
736                ;ITEM 57
737 002104 131601      EM57      ;LKS<07> DOES NOT BECOME 1
738 002106 134621      DH1        ;TEST @, PC
739 002110 136026      DT1        ;$TMP1,$ERRPC
740 002112 000000      0
741                ;ITEM 60
742 002114 131633      EM60      ;WRITE REFERENCE DOESN'T CLEAR LKS<07>
743 002116 134621      DH1        ;TEST @, PC
744 002120 136026      DT1        ;$TMP1,$ERRPC
745 002122 000000      0
746                ;ITEM 61
747 002124 131701      EM61      ;ILLEGAL LKS INTERRUPTS
748 002126 134621      DH1        ;TEST @, PC
749 002130 136026      DT1        ;$TMP1,$ERRPC
750 002132 000000      0
751                ;ITEM 62
752 002134 131730      EM62      ;PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>
753 002136 134621      DH1        ;TEST @, PC
754 002140 136026      DT1        ;$TMP1,$ERRPC
755 002142 000000      0
756                ;ITEM 63
757 002144 132001      EM63      ;LKS READY DOESN'T GO LOW
758 002146 134621      DH1        ;TEST @, PC
759 002150 136026      DT1        ;$TMP1,$ERRPC
760 002152 000000      0
761                ;ITEM 64
762 002154 132032      EM64      ;WRONG NUMBER OF LKS INTERRUPTS
763 002156 134621      DH1        ;TEST @, PC
764 002160 136026      DT1        ;$TMP1,$ERRPC
765 002162 000000      C
766                ;ITEM 65
767 002164 132071      EM65      ;LKS INTERRUPTS HAPPEN AT WRONG PRIORITY
768 002166 135452      DH65      ;TEST @, PC, PRIORITY
769 002170 136262      DT65      ;$TMP1,$ERRPC,$GDDAT
770 002172 000000      0
771                ;ITEM 66
772 002174 132141      EM66      ;BCSR<12> DOES NOT DISABLES LKS
773 002176 134621      DH1        ;TEST @, PC
774 002200 136026      DT1        ;$TMP1,$ERRPC
775 002202 000000      0
776                ;ITEM 67

```

ERROR DEFINITIONS

777	002204	132200	EM67	;BCSR<13> DOESN'T SET LKS<06>
778	002206	134621	DH1	;TEST #, PC
779	002210	136026	DT1	;\$TMP1,\$ERRPC
780	002212	000000	0	
781			;ITEM 70	
782	002214	132235	EM70	;RESET DOESN'T SET LKS<7>
783	002216	134621	DH1	;TEST #, PC
784	002220	136026	DT1	;\$TMP1,\$ERRPC
785	002222	000000	0	
786			;ITEM 71	
787	002224	132266	EM71	;RESET DOESN'T CLEAR LKS<06>
788	002226	134621	DH1	;TEST #, PC
789	002230	136026	DT1	;\$TMP1,\$ERRPC
790	002232	000000	0	
791			;ITEM 72	
792	002234	132322	EM72	;TIMEOUT READING SLU REGISTERS
793	002236	135516	DH72	;TEST #, PC, ADDRESS FAILED
794	002240	136154	DT41	;\$TMP1,\$ERRPC,\$BDDAT
795	002242	000000	0	
796			;ITEM 73	
797	002244	132360	EM73	;XMIT READY DIDN'T GO LOW
798	002246	134621	DH1	;TEST #, PC
799	002250	136026	DT1	;\$TMP1,\$ERRPC
800	002252	000000	0	
801			;ITEM 74	
802	002254	132404	EM74	;RCSR DOESN'T BECOME 1
803	002256	134621	DH1	;TEST #, PC
804	002260	136026	DT1	;\$TMP1,\$ERRPC
805	002262	000000	0	
806			;ITEM 75	
807	002264	132435	EM75	;WRONG CHARACTER RECEIVED
808	002266	134646	DH4	;TEST #, PC, EXPECTED DATA, RECEIVED DATA
809	002270	136272	DT75	;\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
810	002272	000000	0	
811			;ITEM 76	
812	002274	132466	EM76	;RCSR<07> NOT CLEARED AFTER READING RBUF
813	002276	134621	DH1	;TEST #, PC
814	002300	136026	DT1	;\$TMP1,\$ERRPC
815	002302	000000	0	
816			;ITEM 77	
817	002304	132536	EM77	;XCSR<07> NOT SET ON RESET
818	002306	134621	DH1	;TEST #, PC
819	002310	136026	DT1	;\$TMP1,\$ERRPC
820	002312	000000	0	
821			;ITEM 100	
822	002314	132570	EM100	;RCSR<07> NOT CLEARED ON RESET
823	002316	134621	DH1	;TEST #, PC
824	002320	136026	DT1	;\$TMP1,\$ERRPC
825	002322	000000	0	
826			;ITEM 101	
827	002324	132626	EM101	;SLU INTERRUPTS HAPPEN AT 4
828	002326	134621	DH1	;TEST #, PC
829	002330	136026	DT1	;\$TMP1,\$ERRPC
830	002332	000000	0	
831			;ITEM 102	
832	002334	132661	EM102	;RESET DOES NOT CLEAR XCSR<6> AND RCSR<6>
833	002336	134621	DH1	;TEST #, PC

ERROR DEFINITIONS

834	002340	136026	DT1	;\$TMP1,\$ERRPC
835	002342	000000	0	
836			;ITEM 103	
837	002344	132740	EM103	;\$TRANSMIT INTFRRUPT DOES NOT CLEAR XCSR<07>
838	002346	134620	DH1	;\$TEST \$, PC
839	002350	136026	DT1	;\$TMP1,\$ERRPC
840	002352	000000	0	
841			;ITEM 104	
842	002354	133016	EM104	;\$RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>
843	002356	134621	DH1	;\$TEST \$, PC
844	002360	136026	DT1	;\$TMP1,\$ERRPC
845	002362	000000	0	
846			;ITEM 105	
847	002364	133066	EM105	;\$BREAK CONDITION DOES NOT SET RBUF PROPERLY
848	002366	135562	DH105	;\$TEST \$, PC, RBUF
849	002370	136304	DT105	;\$TMP1,\$ERRPC,RBUF
850	002372	000000	0	
851			;ITEM 106	
852	002374	133141	EM106	;\$RBUF WASN'T CLEARED ON NEXT CHAR.
853	002376	135562	DH105	;\$TEST \$, PC, RBUF
854	002400	136304	DT105	;\$TMP1,\$ERRPC,RBUF
855	002402	000000	0	
856			;ITEM 107	
857	002404	133217	EM107	;\$ERROR IN WRITING TO XCSR<0>
858	002406	134621	DH1	;\$TEST \$, PC
859	002410	136026	DT1	;\$TMP1,\$ERRPC
860	002412	000000	0	
861			;ITEM 110	
862	002414	133253	EM110	;\$RESET DOES NOT CLEAR XCSR<00>
863	002416	134621	DH1	;\$TEST \$, PC
864	002420	136026	DT1	;\$TMP1,\$ERRPC
865	002422	000000	0	
866			;ITEM 111	
867	002424	133315	EM111	;\$FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND
868	002426	134646	DH4	;\$TEST \$, PC, EXPECTED DATA, RECEIVED DATA
869	002430	136272	DT75	;\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
870	002432	000000	0	
871			;ITEM 112	
872	002434	133373	EM112	;\$OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF
873	002436	135562	DH105	;\$TEST \$, PC, RBUF
874	002440	136304	DT105	;\$TMP1,\$ERRPC,RBUF
875	002442	000000	0	
876			;ITEM 113	
877	002444	133456	EM113	;\$RBUF WAS NOT CLEARED ON THE NEXT CHARACTER
878	002446	135562	DH105	;\$TEST \$, PC, RBUF
879	002450	136304	DT105	;\$TMP1,\$ERRPC,RBUF
880	002452	000000	0	
881			;ITEM 114	
882	002454	133542	EM114	;\$ERROR IN XCSR<2>
883	002456	134621	DH1	;\$TEST \$, PC
884	002460	136026	DT1	;\$TMP1,\$ERRPC
885	002462	000000	0	
886			;ITEM 115	
887	002464	133563	EM115	;\$ERROR IN TAG STORE FROM STANDALONE MODE
888	002466	135614	DH115	;\$TEST \$, PC, MSER, ADDRESS ACCESSED
889	002470	136314	DT115	;\$TMP1,\$ERRPC,\$BDDAT,\$KIPAR6,\$BDADR
890	002472	000000	0	

ERROR DEFINITIONS

891			:ITEM 116	
892	002474	133633	EM116	;DMA TAG PARITY DOES NOT SET MSER<4>
893	002476	134621	DH1	;TEST #, PC
894	002500	136026	DT1	;\$TMP1,\$ERRPC
895	002502	000000	0	
896			:ITEM 117	
897	002504	133714	EM117	;MSER<13>NOT SET IN STANDALONE MODE
898	002506	134621	DH1	;TEST #, PC
899	002510	136026	DT1	;\$TMP1,\$ERRPC
900	002512	000000	0	
901			:ITEM 120	
902	002514	133757	EM120	;DMA WRITE HITS DON'T INVALIDATE CACHE
903	002516	134621	DH1	;TEST #, PC
904	002520	136026	DT1	;\$TMP1,\$ERRPC
905	002522	000000	0	
906			:ITEM 121	
907	002524	134025	EM121	;IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED
908	002526	134621	DH1	;TEST #, PC
909	002530	136026	DT1	;\$TMP1,\$ERRPC
910	002532	000000	0	
911			:ITEM 122	
912	002534	134123	EM122	;READ DMA HIT IS MESSED UP
913	002536	134621	DH1	;TEST #, PC
914	002540	136026	DT1	;\$TMP1,\$ERRPC
915	002542	000000	0	
916			:ITEM 123	
917	002544	134151	EM123	;ERROR IN DMA CYCLES FROM Q22BE
918	002546	134621	DH1	;TEST #, PC
919	002550	136026	DT1	;\$TMP1,\$ERRPC
920	002552	000000	0	
921			:ITEM 124	
922	002554	134203	EM124	;PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS
923	002556	134621	DH1	;TEST #, PC
924	002560	136026	DT1	;\$TMP1,\$ERRPC
925	002562	000000	0	
926			:ITEM 125	
927	002564	134275	EM125	;NO POWER DOWN TRAP TO 24 OCCUR
928	002566	134621	DH1	;TEST #, PC
929	002570	136026	DT1	;\$TMP1,\$ERRPC
930	002572	000000	0	
931			:ITEM 126	
932	002574	134334	EM126	;ERROR IN INTERRUPTS FROM Q22BE
933	002576	134621	DH1	;TEST #, PC
934	002600	136026	DT1	;\$TMP1,\$ERRPC
935	002602	000000	0	
936			:ITEM 127	
937	002604	134371	EM127	;ERROR IN PMG COUNTER
938	002606	134621	DH1	;TEST #, PC
939	002610	136026	DT1	;\$TMP1,\$ERRPC
940	002612	000000	0	
941			:ITEM 130	
942	002614	134433	EM130	;UNEXPECTED TIMEOUT
943	002616	134621	DH1	;TEST #, PC
944	002620	136330	DT130	;\$TMP1,\$ERRPC
945	002622	000000	0	
946			:ITEM 131	
947	002624	134460	EM131	;ERROR WRITING TO LKS<6>

ERROR DEFINITIONS

```

948 002626 134621          DH1          ;TEST 0, PC
949 002630 136026          DT1          ;$TMP1,$ERRPC
950 002632 000000          0
951                      ;ITEM 132
952 002634 134510          EM132         ;MAINTENANCE REGISTER ERROR
953 002636 134621          DH1          ;TEST 0, PC
954 002640 136026          DT1          ;$TMP1,$ERRPC
955 002642 000000          0
956                      ;ITEM 133
957 002644 134546          EM133         ; EEROM TYPE ERROR
958 002646 134621          DH1          ;TEST 0, PC
959 002650 136026          DT1          ;$TMP1,$ERRPC
960 002652 000000          0
961                      ;ITEM 134
962 002654 134567          EM134         ; ERROM WRITE/READ ERROR
963 002656 135671          DH134        ;TEST 0, PC, ERROR COUNT, PATTERN, DATA READ, ADDRESS
964 002660 136336          DT134        ;$TMP1,$ERRPC,$TMP0,R3,$BDDAT,$BDADR
965 002662 000000          0

```

.SBTTL GLOBAL VARIABLES AND REGISTER NAMES

```

966
967
968
969                      ;REGISTERS FOR THE FIRST Q22BE
970 002664 000000          CSR1: .WORD 0          ;CONTROL REGISTER 1 FOR Q22BE
971 002666 000000          CSR2: .WORD 0          ;CONTROL/STATUS REGISTER 2
972 002670 000000          BA: .WORD 0          ;DMA ADDRESS FOR Q22BE
973 002672 000000          WC: .WORD 0          ;WORD COUNT REGISTER
974 002674 000000          DATA: .WORD 0        ;DMA DATA FOR Q22BE
975 002676 000000          VQBE1: .WORD 0        ;ADDRESS OF VECTOR FOR Q22BE
976 002700 000000          VQPR1: .WORD 0        ;PRIORITY
977 002702 000000          SIMGOA: .WORD 0       ;SIMULTANEUOS GO ADDRFS REGISTER
978
979                      ;REGISTERS FOR THE SECOND Q22BE
980 002704 000000          CSR12: .WORD 0         ;CONTROL REGISTER 1 FOR Q22BE
981 002706 000000          CSR22: .WORD 0        ;CONTROL/STATUS REGISTER 2
982 002710 000000          BA2: .WORD 0         ;DMA ADDRESS FOR Q22BE
983 002712 000000          WC2: .WORD 0         ;WORD COUNT REGISTER
984 002714 000000          DATA2: .WORD 0      ;DMA DATA FOR Q22BE
985 002716 000000          VQBE2: .WORD 0        ;ADDRESS OF VECTOR FOR Q22BE
986 002720 000000          VQPR2: .WORD 0        ;PRIORITY
987
988 002722 000000          LKSFL: .WORD 0
989 002724 000000          ACTCHS: .WORD 0      ;ACTUAL CHECKSUM
990 002726 000000          SAVPCR: .WORD 0
991 002730 000000          SAVBR: .WORD 0
992                      .-2740
993 002740 000000          TEMP: .WORD 0
994 002742          .BLKW 15.          ;RESERVED FOR BLOCK MODE TRANSFER
995 003000 000000          TIMEOUT: .WORD 0
996 003002 000032 000012 000006 Q22EN: .WORD 32,12,6,2 ;PRIORITY 7-4 FOR Q22BE
997 003010 000002
998
999                      BCR=          177524          ;BOOT/DIAGNOSTICS CONFIGURATION
1000                      BDR=          177524          ;BOOT/DIAGNOSTICS DISPLAY
1001                      BCSR=         177520          ;BOOT/DIAGNOSTICS STATUS
1002                      CCR=          177746          ;CACHE CONTROL REGISTER
1003                      HITMIS=        177752          ;HIT OR MISS REGISTER

```

GLOBAL VARIABLES AND REGISTER NAMES

1004	177734	KMCR=	177734	;UNIBUS CONFIGURATION REGISTER
1005	177546	LKS=	177546	;CLOCK STATUS REGISTER
1006	177750	MAIREG=	177750	;MAINTENANCE REGISTER
1007	177744	MSER=	177744	;MEMORY SYSTEM ERROR
1008	177522	PCR=	177522	;PAGE CONTROL REGISTER
1009	177772	PIR=	177772	;PROGRAM INTERRUPT REQUEST
1010	177562	RBUF=	177562	;RECEIVER DATA BUFFER
1011	177560	RCSR=	177560	;RECEIVER STATUS REGISTER
1012	177566	XBUF=	177566	;TRANSMITTER DATA BUFFER
1013	177564	XCSR=	177564	;TRANSMITTER STATUS REGISTER
1014				
1015	177766	CPEREG=	177766	;CPU ERROR REGISTER
1016				
1017	177572	MMRO=SR0		;MEMORY MANAGEMENT REG. 0
1018	177574	MMR1=SR1		;MEMORY MANAGEMENT REG. 1
1019	177576	MMR2=SR2		;MEMORY MANAGEMENT REG. 2
1020	172516	MMR3=SR3		;MEMORY MANAGEMENT REG. 3
1021	120001	POLY= 120001		
1022	000000	NULL=	0	
1023				
1024		.SBTTL	GLOBAL DATA SECTION	
1025				
1026		;	**	
1027		;	THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED	
1028		;	IN MORE THAN ONE TEST.	
1029		;	--	
1030				
1031		;	THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA	
1032		;	WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY	
1033		;	FROM THE DEFAULT RESPONSE.	
1034	003012	000000	SLOC00: .WORD 0	
1035	003014	000000	SLOC01: .WORD 0	
1036				
1037		;	THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.	
1038	003016	000000	LOWADD: .WORD 0	;STORES LOW ADDRESS FOR RAM TESTS
1039	003020	000000	GOODAD: .WORD 0	;STORES GOOD ADDRESS FOR RAM TESTS
1040	003022	000000	ERRCNT: .WORD 0	; CONTAINS TOTAL NO. OF EEROM ERRORS
1041	003024	000000	TSTADD: .WORD 0	;ADDRESS STORE FOR RAM TESTS
1042	003026	000000	NEWADD: .WORD 0	;ADDRESS STORE FOR RAM TEST
1043	003030	000000	FLAG: .WORD 0	;USED TO STORE "FLAG" CONDITIONS
1044	003032	000000	CCHPAS: .WORD 0	; flag-counter for control of Cache subtests
1045	003034	000000	EEPAS: .WORD 0	; flag-counter for control of EEROM subtest
1046	003036	000000	SAVSUP: .WORD 0	;USED TO STORE SUPERVISOR STACK VALUE
1047	003040	000000	SAVUSE: .WORD 0	;USED TO STORE USER STACK VALUE
1048	003042	000000	SAVMRO: .WORD 0	;USED TO STORE MMU STATUS REGISTER 0 DATA
1049	003044	000000	SAVMR1: .WORD 0	;USED TO STORE MMU STATUS REGISTER 1 DATA
1050	003046	000000	SAVMR2: .WORD 0	;USED TO STORE MMU STATUS REGISTER 2 DATA
1051	003050	000000	SAVSWR: .WORD 0	; SAVE SFTWRE SWTCH REG DURING EEROM TEST
1052	003052		FLOAT: .BLKW 4	;USED TO STORE VALUES FOR MMU TESTS
1053	003062		FLO: .BLKW 4	;USED TO STORE VALUES FOR MMU TESTS
1054	003072	000000	SEQ: .WORD 0	;STORES SEQUENCE NUMBER FOR JUMP TESTS
1055	003074	000000	SPS: .WORD 0	;STORES STACK POINTER FOR JUMP TESTS
1056	003076	000000	SPSJ: .WORD 0	;STORES STACK POINTER FOR JUMP TESTS
1057	003100		BTEXP: .BLKW 4	;STORES EXPONENT DURING BIT TESTS
1058	003110		BTRES: .BLKW 4	;STORES RECIEVED DATA FOR BIT TESTS
1059	003120	000000	COUNT: .WORD 0	;ERROR INDICATOR FOR FLOATING POINT TESTS
1060	003122		RECPEC: .BLKW 4	;RECIEVED FLOATING POINT EXCEPTION CODE

GLOBAL DATA SECTION

```

1061 003132          RECST: .BLKW  4          ;RECIEVED FLOATING POINT STATUS
1062 003142          RECDST: .BLKW  4         ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1063
1064                ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1065 003152  000000  ALLCTR: .WORD  0
1066 003154  000000  LOOPIN: .WORD  0
1067
1068                ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1069 003156  000000  SAVPOS: .WORD  0          ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1070 003160  000000  MASK:   .WORD  0          ;STORES BIT MASK FOR ERROR ISOLATION
1071
1072                ;!!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!!!!!
1073 003162  000000  TSTLOC: .WORD  0
1074 003164          .BLKW  20.
1075                ;FPP REGISTER DEFINITIONS
1076                AC0= #0
1077                AC1= #1
1078                AC2= #2
1079                AC3= #3
1080                AC4= #4
1081                AC5= #5
1082                AC6= #6
1083                AC7= #7
1084
1085                ;FPP INTERRUPT VECTOR
1086
1087                FPVEC=244
1088
1089
1090                STBOT= 1000
1091
1092
1094 003234  123456  TAB1:  .WORD  123456
1095 003236  000000  .WORD  000000
1096 003240  000000  .WORD  0
1097 003242  000001  .WORD  1
1098 003244  055555  TAB2:  .WORD  055555
1099 003246  177777  .WORD  177777
1100 003250  145671  .WORD  145671
1101 003252  100000  .WORD  100000
1102 003254  003000  TAB3:  .WORD  003000
1103 003256  123456  .WORD  123456
1104 003260  000000  .WORD  0
1105 003262  000000  .WORD  0
1106 003264  055555  TAB4:  .WORD  55555
1107 003266  177777  .WORD  -1
1108 003270  000000  .WORD  0
1109 003272  000000  .WORD  0
1110 003274  043243  TAB5:  .WORD  43243
1111 003276  000000  .WORD  0
1112 003300  000000  .WORD  0
1113 003302  000000  .WORD  0
1114 003304  162400  TAB5A: .WORD  162400
1115 003306  000000  .WORD  0
1116 003310  000000  .WORD  0
1117 003312  000000  .WORD  0
1118 003314  000000  TAB6:  .WORD  0

```

GLOBAL DATA SECTION

1119	003316	000000				.WORD	0
1120	003320	000000				.WORD	0
1121	003322	000000				.WORD	0
1122	003324	047050			TAB6A:	.WORD	47050
1123	003326	010000				.WORD	10000
1124	003330	000000				.WORD	0
1125	003332	000000				.WORD	0
1126	003334	000200			TAB7:	.WORD	200
1127	003336	000000				.WORD	0
1128	003340	000000				.WORD	0
1129	003342	000000				.WORD	0
1130	003344	000200			TAB8:	.WORD	200
1131	003346	000000				.WORD	0
1132	003350	000000				.WORD	0
1133	003352	000001				.WORD	1
1134	003354	000400	000000	000000	TAB9:	.WORD	400,0,0,0
	003362	000000					
1135	003364	030000			TAB10:	.WORD	30000
1136	003366	003000				.WORD	3000
1137	003370	000000				.WORD	0
1138	003372	000000				.WORD	0
1139	003374	016400			TAB11:	.WORD	16400
1140	003376	000000				.WORD	0
1141	003400	000000				.WORD	0
1142	003402	000000				.WORD	0
1143	003404	030000	003000	000002	TAB11A:	.WORD	30000,3000,2,0
	003412	000000					
1144	003414	016100	000000	000000	TAB12:	.WORD	16100,0,0,1
	003422	000001					
1145	003424	016200			TAB13:	.WORD	16200
1146	003426	000000				.WORD	0
1147	003430	000000				.WORD	0
1148	003432	000001				.WORD	1
1149	003434	030000	003000	000000	TAB13B:	.WORD	30000,3000,0,140000
	003442	140000					
1150	003444	030000			TAB14:	.WORD	30000
1151	003446	000000				.WORD	0
1152	003450	000000				.WORD	0
1153	003452	000000				.WORD	0
1154	003454	024700			TAB15:	.WORD	24700
1155	003456	000000				.WORD	0
1156	003460	000000				.WORD	0
1157	003462	000000				.WORD	0
1158	003464	025000			TAB16:	.WORD	25000
1159	003466	175363				.WORD	175363
1160	003470	123456				.WORD	123456
1161	003472	123456				.WORD	123456
1162	003474	030000			TAB17:	.WORD	30000
1163	003476	007020				.WORD	7020
1164	003500	000000	000000			.WORD	0,0
1165	003504	023456			TAB18:	.WORD	23456
1166	003506	000000				.WORD	0
1167	003510	000000				.WORD	0
1168	003512	000001				.WORD	1
1169	003514	100200	000000	000000	TAB21:	.WORD	100200,0,0,0
	003522	000000					
1170	003524	100400	000000	000000	TAB22:	.WORD	100400,0,0,0

GLOBAL DATA SECTION

1171	003532	000000					
	003534	000200	000000	000000	TAB23:	.WORD	200,0,0,1
	003542	000001					
1172	003544	062400	000000	000000	TAB24:	.WORD	62400,0,0,0
	003552	000000					
1173	003554	001100	000000	000000	TAB25:	.WORD	1100,0,0,0
	003562	000000					
1174	003564	100600	000000	000000	TAB26:	.WORD	100600,0,0,0
	003572	000000					
1175	003574	001000	000000	000000	TAB27:	.WORD	1000,0,0,0
	003602	000000					
1176	003604	000600	000000	000000	TAB28:	.WORD	600,0,0,0
	003612	000000					
1177	003614	010100	000000	000000	TAB29:	.WORD	10100,0,0,0
	003622	000000					
1178	003624	010100	000000	002000	TAB29A:	.WORD	10100,0,2000,0
	003632	000000					
1179							
1180	003634	000500	000000	000000	TAB30:	.WORD	500,0,0,0
	003642	000000					
1181	003644	100400	000000	000000	TAB31:	.WORD	100400,0,0,0
	003652	000000					
1182	003654	016000	000000	000000	TAB32:	.WORD	16000,0,0,0
	003662	000000					
1183	003664	011600	000000	000000	TAB33:	.WORD	11600,0,0,0
	003672	000000					
1184	003674	000640	000000	000000	TAB34:	.WORD	640,0,0,0
	003702	000000					
1185	003704	077600	000000	000000	TAB40:	.WORD	77600,0,0,0
	003712	000000					
1186	003714	100200	000000	000000	TAB41:	.WORD	100200,0,0,1
	003722	000001					
1187	003724	000340	000000	000000	TAB42:	.WORD	340,0,0,0
	003732	000000					
1188	003734	000077	177777	177777	TAB43:	.WORD	77,177777,177777,177776
	003742	177776					
1189	003744	000577	177777	177777	TAB45:	.WORD	577,-1,-1,-1
	003752	177777					
1190	003754	000577	177777	000000	TAB46:	.WORD	577,-1,0,0
	003762	000000					
1191	003764	173737	124242	052525	TAB47:	.WORD	173737,124242,052525,12346
	003772	012346					
1192	003774	000000	000000	052525	TAB47A:	.WORD	0,0,052525,12346
	004002	012346					
1193	004004	173737	124242	000000	TAB48:	.WORD	173737,124242,0,0
	004012	000000					
1194	004014	000600	000000	000000	TAB49:	.WORD	600,0,0,0
	004022	000000					

1195

1196 004024

START:

:: LCP/ORION ROUTINE TO SAVE EMULATOR AND PRIORITY

004024	005737	004112			EMTSAV: TST	SAV30	:: FIRST TIME THROUGH ?
004030	001034				BNE	VMKOR	:: BRANCH IF BEEN HERE ALREADY
004032	032737	000040	000052		BIT	#BIT5,#52	:: ARE WE IN UFD MODE ?
004040	001430				BEQ	VMKOR	:: LEAVE IF NOT
004042	012737	177777	004116		MOV	#-1,UFD FLG	:: SET UFD FLAG
004050	032737	000100	000052		BIT	#BIT6,#52	:: ARE WE IN QUIET MODE ?

GLOBAL DATA SECTION

```

004056 001403          BEQ      1$          ;; BR IF NOT
004060 012737 177777 004120      MOV      #-1,UQUIET      ;; SET QUIET MODE
004066 104042          EMT      42          ;; GET ADDRESS OF XXDP DCA TABLE
004070 005060 000042          CLR      42(R0)        ;; CLR XXDP+ "DRSERR"
004074 013737 000030 004112      MOV      30,SAV30       ;; SAVE EMULATOR ADDRESS
004102 013737 000032 004114      MOV      32,SAV32       ;; SAVE EMULATOR PRIORITY LEVEL
004110 000404          BR       VMKOR          ;; GET AROUND TAG AREA
004112 000000          SAV30: .WORD 0      ;; PUT EMULATOR INFO HERE
004114 000000          SAV32: .WORD 0      ;; PUT PRIORITY LOCATION      HERE
004116 000000          UFDPLG: .WORD 0     ;; USER FRIENDLY MODE FLAG
004120 000000          UQUIET: .WORD 0    ;; UFD QUIET MODE FLAG
004122          VMKOR:
;*****
1197 004122          1$:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS (#CMTAG) AREA
004122 012706 001100      MOV      #CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
004126 005026          CLR      (R6)+          ;;CLEAR MEMORY LOCATION
004130 022706 001140      CMP      #SWR,R6      ;;DONE?
004134 001374          BNE     -6            ;;LOOP BACK IF NO
004136 012706 001100      MOV      #STACK,SP    ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
004142 012737 140474 000020      MOV      #SCOPE,#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004150 012737 000340 000022      MOV      #340,#IOTVEC+2 ;;LEVEL 7
004156 012737 140776 000030      MOV      #ERROR,#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
004164 012737 000340 000032      MOV      #340,#EMTVEC+2 ;;LEVEL 7
004172 012737 143522 000034      MOV      #TRAP,#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
004200 012737 000340 000036      MOV      #340,#TRAPVEC+2;LEVEL 7
004206 012737 143604 000024      MOV      #PWRDN,#PWRVEC ;;POWER FAILURE VECTOR
004214 012737 000340 000026      MOV      #340,#PWRVEC+2 ;;LEVEL 7
004222 013737 140406 140400      MOV      $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
004230 005037 001164          CLR      $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
004234 005037 001166          CLR      $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
004240 112737 000001 001115      MOV      #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
004246 012737 004246 001106      MOV      #,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
004254 012737 004254 001110      MOV      #,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
004262 013746 000004          MOV      #ERRVEC,-(SP) ;;SAVE ERROR VECTOR
004266 012737 004322 000004      MOV      #30000,$ERRVEC ;;SET UP ERROR VECTOR
004274 012737 177570 001140      MOV      #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
004302 012737 177570 001142      MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
004310 022777 177777 174622      CMP      #-1,$SWR     ;;TRY TO REFERENCE HARDWARE SWR
004316 001012          BNE     30002$       ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = 1
004320 000403          BR       30001$       ;;BRANCH IF NO TIMEOUT
004322 012716 004330      30000$: MOV      #30001$,(SP)  ;;SET UP FOR TRAP RETURN
004326 000002          RTI
004330 012737 000176 001140      30001$: MOV      #SWREG,SWR   ;;POINT TO SOFTWARE SWR
004336 012737 000174 001142      MOV      #DISPREG,DISPLAY
004344 012637 000004      30002$: MOV      (SP)+,$ERRVEC ;;RESTORE ERROR VECTOR
004350 005037 001206          CLR      $PASS        ;;CLEAR PASS COUNT
004354 132737 000200 001221      BITB    #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
004362 001403          BEQ     30003$       ;;YES,USE NON-APT SWITCH
004364 012737 001222 001140      MOV      #SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
004372          30003$.
1198 004372 013737 004116 004120      MOV      UFD LG,UQUIET ;;ABORT IN UFD ON ERROR

```

INITIALIZE THE COMMON TAGS

```

1199 004400 012737 137542 000004      MOV    #TOUT,#ERRVEC ;POINT TO TIMEOUT ROUTINE
1200 004406 012737 00340 000006      MOV    #340,#ERRVEC+2 ;AT PRIORITY 7
1201 004414 012737 137012 000114      MOV    #RAMPAR,#114 ;POINT PARITY ABORT
1202 004422 012737 00340 000116      MOV    #340,#116 ;AT PRIORITY7
1203 004430 012737 137724 000250      MOV    #MMUTRP,#250 ;POINT MMU TRAP VECTOR
1204 004436 012737 000340 000252      MOV    #340,#252 ;
1205 004444 005037 177766 ;CLEAR CPU ERROR REGISTER
1206 004450 032737 000100 000052      BIT    #BIT06,#52 ;IN UFD QUIET MODE ?
1207 004456 001130 ;IF SO, SKIP PRINTOUT
1208 004460 012701 004472      MOV    #.+12,R1 ;STORE LOCATION POINTER
1209 004464 012711 177777      MOV    #-1,(R1) ;MAKE SURE TO PRINT NOT ONLY AT FIRST TIME
1210 ;
      .SBTTL TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
      INC    #-1 ;;FIRST TIME?
      BNE    30004$ ;;BRANCH IF NO
      CMP    #ENDAD #42 ;;ACT-11?
      BEQ    30004$ ;;BRANCH IF YES
      TYPE    ,30005$ ;;TYPE ASCIZ STRING
      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
      TST    #42 ;;ARE WE RUNNING UNDER XXDP/ACT?
      BNE    30006$ ;;BRANCH IF YES
      CMPB   $ENV,#1 ;;ARE WE RUNNING UNDER APT?
      BEQ    30006$ ;;BRANCH IF YES
      CMP    SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
      BNE    30007$ ;;BRANCH IF NO
      GTSWR ;;GET SOFT-SWR SETTINGS
      BR     30007$
      MOVB   #1,$AUTOB ;;SET AUTO-MODE INDICATOR
      BR     30007$
      BR     30004$ ;;GET OVER THE ASCIZ
      ;;30005$: .ASCIZ <CRLF>* KDJ11 B CPU DIAGNOSTIC - COKDAEO *<CRLF>
      30004$:
1211 004622 000240      NOP ; debug aid
1212 004624 123727 001220 000001      CMPB   $ENV,#1 ; running under APT?
1213 004632 001020      BNE    20$ ; default no-APT initialization
1214 004634 013700 001224      MOV    $USWR,R0 ; work copy pass calculation
1215 004640 005700      TST    R0 ; if = 0 default value
1216 004642 001420      BEQ    25$ ; setup default value
1217 ;
1218 004644 042700 000017      BIC    #17,R0 ; clear low order nibble
1219 004650 000241      CLC ; assure no unknowns
1220 004652 006000      ROR    R0 ; 4 rotates = divide
1221 004654 006000      ROR    R0 ; by 16 (=pass time)
1222 004656 006000      ROR    R0 ; this area subroutined
1223 004660 006000      ROR    R0 ; with general purpose
1224 004662 005700      TST    R0 ; oivide, this test to
1225 004664 001413      BEQ    30$ ; determine skip altogether
1226 ;
1227 004666 010037 003032      MOV    R0,CCHPAS ;residue = no. of desired passes
1228 004672 000413      BR     35$ ; continue on
1229 004674 012737 177777 003032 20$: MOV    #-1,CCHPAS ; largest number no apt mode??
1230 004702 000407      BR     35$
1231 004704 012737 000001 003032 25$: MOV    #1,CCHPAS ; normal default = 1
1232 004712 000403      BR     35$
1233 004714 012737 000000 003032 30$: MOV    #0,CCHPAS ; no cache tests included
1234 004722 000240      35$: nop ; debug aid
1235 ;

```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

1236 004724 032737 000200 000052      BIT      @BIT07,@052      ;UFD MODE?
1237 004732 001002                      BNE      LOOP            ;IF YES, BRANCH
1238 004734 004737 137070              JSR      PC,Q22SIZ       ;SIZE FOR Q22BE
1239 004740                          LOOP:
1240                          .DSABLE AMA
1242                          ;*****
004740 000004      TST1:  SCOPE
1243                          .SBTTL  BASE INSTRUCTION SET TESTS
1244                          ;*****
1245                          ;*****
1246                          ;          BEGIN BASE INSTRUCTION SET TESTING
1247                          ;*****
1248                          ;*****
1257 004742      FRSTST:
1258
1259                          ;TEST BEQ BNE INSTRUCTIONS
1260                          ;*****
                                ;THESE TWO INSTRUCTIONS ARE FUNDAMENTAL TO RECOGNIZING ERROR CONDITIONS
1261 004742 000277      SCC
1262 004744 000244      CLZ                      ;CC=0100 - Z BIT CLEARED
1263 004746 001401      BEQ      1$                ;*TEST INSTR - TRY TO CAUSE A BEQ ERROR
1264 004750 001001      BNE      2$                ;BRANCH IF GOOD
1265                          ;THE Z FLAG DIDNT CLEAR OR BRANCH FAILED.
1266                          ;FAILURE AT THIS LOCATION
1267                          ;COULD MEAN A BUS PROBLEM, MICRO CODE PROBLEM
1268                          ;CONDITION CODE PROBLEM OR JUST ABOUT ANYTHING
1269                          ;ELSE.
1270 004752 104001      1$:  ERROR  +1                ;CPU ERROR
1271 004754 000257      2$:  CCC
1272 004756 000264      SEZ                      ;COND CODES = 0100 (ZERO)
1273 004760 001001      BNE      3$                ;*TEST INSTR* TRY TO BRANCH ON ZERO FLAG
1274 004762 001401      BEQ      4$                ;*TEST INSTR* BRANCH IF GOOD
1275                          ;BRANCH FAILURE WITH Z BIT SET
1276 004764 104001      3$:  ERROR  +1                ;CPU ERROR
1277 004766                          4$:
1287 004766                          M2:
1288
1289                          ;          TEST BRANCH ON CARRY
1290                          ;*****
                                ;THIS IS A TEST TO SEE IF THE MODULE FORM ANTICIPATED IS FEASIBLE.
1291 004766 000257      CCC                      ;CC=0000
1292 004770 103001      BCC      2$                ;*TEST INSTR*
1293                          ;BRANCH CARRY CLEAR FAILED
1294 004772 104001      1$:  ERROR  +1                ;CPU ERROR
1295 004774 000261      2$:  SEC
1296 004776 103401      BCS      4$                ;*TEST INSTR*
1297                          ; BRANCH CARRY SET FAILED
1298 005000 104001      3$:  ERROR  +1                ;CPU ERROR
1299 005002                          4$:
1300
1303 005002      M3:
1304
1305                          ;          TEST DATA PATHS
1306 005002 005000      CLR      RO
1307 005004 001005      BNE      1$                ;TRY TO INSURE WE ARE TESTING
1308                          ;THE DATA PATH AND NOT THE "CLR RO" INSTRUCTION
1309 005006 005010      CLR      (RO)              ;FORCE LOCATION TO ZERO

```


BASE INSTRUCTION SET TESTS

```

1375 005156 001401      BEQ      4#           ;YES GO ON
1376                    ;NO GO TO ERROR
1377 005160 104001      ERROR    +1           ;CPU ERROR
1378 005162            4#:
1379                    ;
1382 005162            ;GPR1TS:
1383                    ;
1384 005162 012701 177777  R1 BIT TESTS
1385 005166 020127 177777  MOV      #177777,R1      ;R1=177777
1386 005172 001401      CMP      R1,#177777     ;DOES R1=177777
1387                    BEQ      1#           ;YES GO ON
1388 005174 104001      ERROR    +1           ;NO GO TO ERROR
1389 005176 005001      ;CPU ERROR
1390 005200 020127 000000 1#:  CLR      R1           ;R1=0
1391 005204 001401      CMP      R1,#0         ;DOES R1=0
1392                    BEQ      2#           ;YES GO ON
1393 005206 104001      ERROR    +1           ;NO GO TO ERROR
1394 005210 012701 125252 2#:  MOV      #125252,R1     ;CPU ERROR
1395 005214 020127 125252  CMP      R1,#125252    ;R1=125252
1396 005220 001401      BEQ      3#           ;DOES R1=125252
1397                    ;YES GO ON
1398 005222 104001      ERROR    +1           ;NO GO TO ERROR
1399 005224 012701 052525 3#:  MOV      #52525,R1     ;CPU ERROR
1400 005230 020127 052525  CMP      R1,#52525    ;R1=52525
1401 005234 001401      BEQ      4#           ;DOES R1=52525
1402                    ;YES GO ON
1403 005236 104001      ERROR    +1           ;NO GO TO ERROR
1404 005240            ;CPU ERROR
1405                    4#:
1408 005240            ;GPR2TS:
1409                    ;
1410 005240 012702 177777  R2 BIT TESTS
1411 005244 020227 177777  MOV      #177777,R2    ;R2=177777
1412 005250 001401      CMP      R2,#177777    ;DOES R2=177777
1413                    BEQ      1#           ;YES GO ON
1414 005252 104001      ERROR    +1           ;NO GO TO ERROR
1415 005254 005002      ;CPU ERROR
1416 005256 020227 000000 1#:  CLR      R2           ;R2=0
1417 005262 001401      CMP      R2,#0         ;DOES R2=0
1418                    BEQ      2#           ;YES GO ON
1419 005264 104001      ERROR    +1           ;NO GO TO ERROR
1420 005266 012702 125252 2#:  MOV      #125252,R2    ;CPU ERROR
1421 005272 020227 125252  CMP      R2,#125252    ;R2=125252
1422 005276 001401      BEQ      3#           ;DOES R2=125252
1423                    ;YES GO ON
1424 005300 104001      ERROR    +1           ;NO GO TO ERROR
1425 005302 012702 052525 3#:  MOV      #52525,R2     ;CPU ERROR
1426 005306 020227 052525  CMP      R2,#52525    ;R2=52525
1427 005312 001401      BEQ      4#           ;DOES R2=52525
1428                    ;YES GO ON
1429 005314 104001      ERROR    +1           ;NO GO TO ERROR
1430 005316            ;CPU ERROR
1431                    4#:
1434 005316            ;GPR3TS:
1435                    ;
1436 005316 012703 177777  R3 BIT TESTS
1437 005322 020327 177777  MOV      #177777,R3    ;R3=177777
                    CMP      R3,#177777    ;DOES R3=177777

```

BASE INSTRUCTION SET TESTS

```

1438 005326 001401      BEQ      1#           ;YES GO ON
1439                      ;NO GO TO ERROR
1440 005330 104001      ERROR    +1          ;CPU ERROR
1441 005332 005003      1#:     CLR      R3           ;R3=0
1442 005334 020327 000000  CMP      R3,#0       ;DOES R3=0
1443 005340 001401      BEQ      2#           ;YES GO ON
1444                      ;NO GO TO ERROR
1445 005342 104001      ERROR    +1          ;CPU ERROR
1446 005344 012703 125252  2#:     MOV      #125252,R3 ;R3=125252
1447 005350 020327 125252  CMP      R3,#125252 ;DOES R3=125252
1448 005354 001401      BEQ      3#           ;YES GO ON
1449                      ;NO GO TO ERROR
1450 005356 104001      ERROR    +1          ;CPU ERROR
1451 005360 012703 052525  3#:     MOV      #52525,R3 ;R3=52525
1452 005364 020327 052525  CMP      R3,#52525 ;DOES R3=52525
1453 005370 001401      BEQ      4#           ;YES GO ON
1454                      ;NO GO TO ERROR
1455 005372 104001      ERROR    +1          ;CPU ERROR
1456 005374      4#:
1457                      ;
1460 005374      ;GPR4TS:
1461                      ;
1462 005374 012704 177777      ; R4 BIT TESTS
1463 005400 020427 177777      MOV      #177777,R4 ;R4=177777
1464 005404 001401      CMP      R4,#177777 ;DOES R4=177777
1465                      BEQ      1#           ;YES GO ON
1466 005406 104001      ;NO GO TO ERROR
1467 005410 005004      ERROR    +1          ;CPU ERROR
1468 005412 020427 000000  1#:     CLR      R4           ;R4=0
1469 005416 001401      CMP      R4,#0       ;DOES R4=0
1470                      BEQ      2#           ;YES GO ON
1471                      ;NO GO TO ERROR
1472 005420 104001      ERROR    +1          ;CPU ERROR
1473 005422 012704 125252  2#:     MOV      #125252,R4 ;R4=125252
1474 005426 020427 125252  CMP      R4,#125252 ;DOES R4=125252
1475                      BEQ      3#           ;YES GO ON
1476                      ;NO GO TO ERROR
1477 005434 104001      ERROR    +1          ;CPU ERROR
1478 005436 012704 052525  3#:     MOV      #52525,R4 ;R4=52525
1479 005442 020427 052525  CMP      R4,#52525 ;DOES R4=52525
1480                      BEQ      4#           ;YES GO ON
1481                      ;NO GO TO ERROR
1482 005450 104001      ERROR    +1          ;CPU ERROR
1483                      4#:
1486 005452      ;GPR5TS:
1487                      ;
1488 005452 012705 177777      ; R5 BIT TESTS
1489 005456 020527 177777      MOV      #177777,R5 ;R5=177777
1490 005462 001401      CMP      R5,#177777 ;DOES R5=177777
1491                      BEQ      1#           ;YES GO ON
1492                      ;NO GO TO ERROR
1493 005464 104001      ERROR    +1          ;CPU ERROR
1494 005466 005005      1#:     CLR      R5           ;R5=0
1495 005470 020527 000000  CMP      R5,#0       ;DOES R5=0
1496 005474 001401      BEQ      2#           ;YES GO ON
1497                      ;NO GO TO ERROR
1498 005476 104001      ERROR    +1          ;CPU ERROR
1499 005500 012705 125252  2#:     MOV      #125252,R5 ;R5=125252

```

BASE INSTRUCTION SET TFSTS

```

1499 005504 020527 125252      CMP      R5,#125252      ;DOES R5=125252
1500 005510 001401              BEQ      3$              ;YES GO ON
1501                          ;NO GO TO ERROR
1502 005512 104001              ERROR    +1              ;CPU ERROR
1503 005514 012705 052525      3$:      MOV      #52525,R5    ;R5=52525
1504 005520 020527 052525      CMP      R5,#52525      ;DOES R5=52525
1505 005524 001401              BEQ      4$              ;YES GO ON
1506                          ;NO GO TO ERROR
1507 005526 104001              ERROR    +1              ;CPU ERROR
1508 005530      4$:
1509                          ;
1512 005530      ;GPR6TS:
1513                          ;
1514 005530 012706 177777      ; R6 BIT TESTS
1515 005534 020627 177777      MOV      #177777,R6     ;R6=177777
1516 005540 001401              CMP      R6,#177777     ;DOES R6=177777
1517                          BEQ      1$              ;YES GO ON
1518                          ;NO GO TO ERROR
1518 005542 104001              ERROR    +1              ;CPU ERROR
1519 005544 005006      1$:      CLR      R6              ;R6=0
1520 005546 020627 000000      CMP      R6,#0          ;DOES R6=0
1521 005552 001401              BEQ      2$              ;YES GO ON
1522                          ;NO GO TO ERROR
1523 005554 104001              ERROR    +1              ;CPU ERROR
1524 005556 012706 125252      2$:      MOV      #125252,R6     ;R6=125252
1525 005562 020627 125252      CMP      R6,#125252     ;DOES R6=125252
1526 005566 001401              BEQ      3$              ;YES GO ON
1527                          ;NO GO TO ERROR
1528 005570 104001              ERROR    +1              ;CPU ERROR
1529 005572 012706 052525      3$:      MOV      #52525,R6     ;R6=52525
1530 005576 020627 052525      CMP      R6,#52525     ;DOES R6=52525
1531 005602 001401              BEQ      4$              ;YES GO ON
1532                          ;NO GO TO ERROR
1533 005604 104001              ERROR    +1              ;CPU ERROR
1534 005606 012706 001000      4$:      MOV      #STBOT,R6     ;RESTORE SP
1535                          ;
1536                          ;PSWBTS:
1539 005612      ;
1540                          ;
1541 005612 012737 000377 177776  ; PSW LOW BYTE BIT TESTS
1542 005620 022737 000357 177776  MOV      #377,#177776   ;PS=357 T BIT SHOULDN'T SET
1543 005626 001401              CMP      #357,#177776   ;DOES PS=357
1544                          BEQ      1$              ;YES GO ON
1545                          ;NO GO TO ERROR
1545 005630 104001              ERROR    +1              ;CPU ERROR
1546 005632 005037 177776      1$:      CLR      #177776        ;PS=0
1547 005636 022737 000000 177776  CMP      #0,#177776     ;DOES PS=0
1548 005644 001401              BEQ      2$              ;YES GO ON
1549                          ;NO GO TO ERROR
1550 005646 104001              ERROR    +1              ;CPU ERROR
1551 005650 012737 000105 177776  2$:      MOV      #105,#177776   ;PS=105
1552 005656 022737 000105 177776  CMP      #105,#177776   ;DOES PS=105
1553 005664 001401              BEQ      3$              ;YES GO ON
1554                          ;NO GO TO ERROR
1555 005666 104001              ERROR    +1              ;CPU ERROR
1556 005670 012737 000252 177776  3$:      MOV      #252,#177776   ;PS=252
1557 005676 022737 000252 177776  CMP      #252,#177776   ;DOES PS=252
1558 005704 001401              BEQ      4$              ;YES GO ON
1559                          ;NO GO TO ERROR

```

BASE INSTRUCTION SET TESTS

```

1560 005706 104001          ERROR +1          ;CPU ERROR
1561 005710          4$:
1571
1572 005710          MSP0:
1573
1574          ;      TEST SINGLE OPERAND INSTRUCTIONS- MODE 0
1575          ;*****
          ;THE INC, COM, CLR, AND DECREMENT INSTRUCTIONS ARE VERIFIED.
          CLR      R4          ;INITIALIZE R4 WITH DATA
          COM      R4          ;
          CLR      R4          ;*TEST INSTRUCTION
          BEQ      2$          ;BRANCH IF R4 CLEARED
1576 005710 005004          1$:          ERROR +1          ;CPU ERROR
1577 005712 005104          2$:          COM      R4          ;*TEST COMPLIMENT INSTRUCTION
1578 005714 005004          INC      R4          ;*TEST INCREMENT INSTRUCTION
1579 005716 001401          BEQ      4$          ;BRANCH IF R4 =0
1580 005720 104001          ;COMPLIMENT OR INCREMENT FAILED
1581 005722 005104          3$:          ERROR +1          ;CPU ERROR
1582 005724 005204          4$:
1583 005726 001401          ;
1584
1585 005730 104001          MSPB:
1586 005732
1587
1588          ;
1591 005732          ;      TEST SINGLE OPS - EVEN BYTE OF CLRB, DECB, AND COMB
1592          CLR      R4
1593          COM      R4          ;SETUP TEST REGISTER
1594 005732 005004          CLRB     R4          ;*TEST CLEAR BYTE INSTRUCTION
1595 005734 005104          BEQ      2$          ;BRANCH IF GOOD
1596 005736 105004          ;CLEAR EVEN BYTE FAILED
1597 005740 001401          1$:          ERROR +1          ;CPU ERROR
1598
1599 005742 104001          2$:          DECB     R4          ;*TEST DECREMENT BYTE
1600 005744 105304          BPL      3$          ;DECREMENT BYTE FAILED
1601 005746 100002          COMB     R4          ;*TEST COMPLIMENT BYTE
1602 005750 105104          BEQ      4$          ;BRANCH IF GOOD
1603 005752 001401          ;COMPLIMENT OR DECREMENT FAILED TO WORK
1604
1605 005754 104001          3$:          ERROR +1          ;CPU ERROR
1606 005756          4$:
1607
1608          ;
1611 005756          MSPC:
1612          ;      TEST SINGLE OPS - MODE 1 CLRB, COMB, AND INCB
1613 005756 005004          CLR      R4
1614 005760 005014          CLR      (R4)
1615 005762 005114          COM      (R4)          ;SETUP TEST DATA
1616 005764 005014          CLR      (R4)          ;*TEST INSTRUCTION
1617 005766 001401          BEQ      2$          ;BRANCH IF GOOD
1618
1619 005770 104001          1$:          ERROR +1          ;CPU ERROR
1620 005772 005114          2$:          COM      (R4)          ;*TEST INSTRUCTION
1621 005774 001403          BEQ      3$          ;(0)SHOULD = -1
1622 005776 100002          BPL      3$          ;
1623 006000 005214          INC      (R4)          ;*TEST INSTRUCTION
1624 006002 001401          BEQ      4$          ;BRANCH IF GOOD
1625
1626 006004 104001          3$:          ERROR +1          ;COM OR INC FAILED TO ALTER LOC 0 CORRECTLY
1627 006006          4$:          ;CPU ERROR
1628

```


BASE INSTRUCTION SET TESTS

```

1629
1632 006006      ; MSPD:
1633
1634      ; TEST SINGLE OPS MODE1-EVEN BYTE-CLRB,COMB,INCB
1635 006006 005004 CLR R4
1636 006010 005014 CLR (R4)
1637 006012 005114 COM (R4) ; SETUP TEST DATA
1638 006014 105014 CLRB (R4) ; *TEST INSTRUCTION
1639 006016 105014 CLRB (R4) ; *TEST INSTRUCTION
1640 006020 001401 BEQ 2$ ; BRANCH IF GOOD
1641 ; CLEAR (0) EVEN BYTE FAILED
1642 006022 104001 1$: ERROR +1 ; CPU ERROR
1643 006024 105214 2$: INCB (R4) ; *TEST INSTRUCTION
1644 006026 100405 BMI 3$ ; TEST FLAGS
1645 006030 001404 BEQ 3$
1646 006032 105114 COMB (R4) ; *TEST INSTRUCTION
1647 006034 105214 INCB (R4)
1648 006036 105214 INCB (R4)
1649 006040 001401 BEQ 4$ ; BRANCH IF GOOD
1650 ; COMB OR INCB FAILED
1651 006042 104001 3$: ERROR +1 ; CPU ERROR
1652 006044 4$:
1653
1654      ; MSPEO:
1657 006044
1658
1659      ; TEST SINGLE OPS - ODD BYTE - CLRB, COMB, DECB
1660 006044 005004 CLR R4
1661 006046 005014 CLR (R4)
1662 006050 005114 COM (R4) ; SETUP TEST DATA
1663 006052 005204 INC R4 ; POINT TO ODD BYTE
1664 006054 105014 CLRB (R4) ; *TEST INSTRUCTION
1665 006056 001401 BEQ 1$ ; BRANCH IF GOOD
1666 ; CLEAR ODD BYTE FAILED
1667 006060 104001 ERROR +1 ; CPU ERROR
1668 006062 005304 1$: DEC R4 ; POINT TO EVEN BYTE
1669 006064 005214 INC (R4) ; LOC 0=1 0
1670 006066 005204 INC R4 ; POINT TO ODD BYTE
1671 006070 105114 COMB (R4) ; *TEST INSTRUCTION
1672 006072 105214 INCB (R4) ; LOC 0=-1 0
1673 006074 100003 BPL 2$ ; BRANCH IF ERROR
1674 006076 001402 BEQ 2$ ;
1675 006100 105214 INCB (R4) ; *TEST INSTRUCTION
1676 006102 001401 BEQ 3$ ; BRANCH IF GOOD
1677 ; MODE 1, ODD BYTE FAILED
1678 006104 104001 2$: ERROR +1 ; CPU ERROR
1679 006106 3$:
1680
1681      ; MSPF:
1684 006106
1685      ; TEST SINGLE OP MODE 2 - CLR, COM, INC
1686 006106 005004 CLR R4
1687 006110 105104 COMB R4
1688 006112 005204 INC R4 ; R4=400
1689 006114 005014 CLR (R4) ; 400=0
1690 006116 005114 COM (R4) ; 400=-1
1691 006120 005024 CLR (R4) ; *TEST INSTRUCTION

```

BASE INSTRUCTION SET TESTS

```

1692 006122 001401      BEQ      1$      ;BRANCH IF GOOD
1693                                     ;MODE 2 CLEAR FAILED
1694 006124 104001      ERROR    +1      ;CPU ERROR
1695 006126 005304      1$:      DEC      R4
1696 006130 005304      DEC      R4      ;R4=400
1697 006132 005124      COM      (R4)+   ;*TEST INSTRUCTION
1698 006134 100004      BPL      2$      ;BRANCH IF FAILURE
1699 006136 005304      DEC      R4
1700 006140 005304      DEC      R4      ;R4=400
1701 006142 005224      INC      (R4)+   ;*TEST INSTRUCTION
1702 006144 001401      BEQ      3$      ;BRANCH IF GOOD
1703                                     ;MODE 2 FAILURE
1704 006146 104001      2$:      ERROR    +1      ;CPU ERROR
1705 006150      3$:
1706
1707      ;
1710 006150      MSPG:
1711
1712      ;      TEST CLRB, COMB, DECB, MODE 2 - EVEN BYTE
1713 006150 005004      CLR      R4
1714 006152 105104      COMB     R4
1715 006154 005204      INC      R4      ;R4=400
1716 006156 005014      CLR      (R4)
1717 006160 005114      COM      (R4)   ;400=-1
1718 006162 105024      CLRB     (R4)+  ;*TEST INSTRUCTION
1719 006164 001401      BEQ      1$      ;BRANCH IF GOOD
1720                                     ;MODE 2 EVEN BYTE FAILED
1721 006166 104001      ERROR    +1      ;CPU ERROR
1722 006170 005304      1$:      DEC      R4
1723 006172 105324      DECB     (R4)+  ;*TEST INSTRUCTION
1724 006174 100003      BPL      2$      ;BRANCH IF BAD
1725 006176 005304      DEC      R4      ;POINT TO EVEN BYTE
1726 006200 105124      COMB     (R4)+  ;*TEST INSTRUCTION
1727 006202 001401      BEQ      3$      ;BRANCH IF GOOD
1728                                     ;MODE 2, EVEN BYTE FAILED
1729 006204 104001      2$:      ERROR    +1      ;CPU ERROR
1730 006206      3$:
1731
1732      ;
1735 006206      MSPH:
1736
1737      ;      TEST CLRB, COMB, INCB MODE 2 - ODD BYTE
1738 006206 005004      CLR      R4
1739 006210 105104      COMB     R4
1740 006212 005204      INC      R4      ;R4=400
1741 006214 005014      CLR      (R4)
1742 006216 005114      COM      (R4)   ;400=-1 1
1743 006220 005214      INC      (R4)   ;POINT TO ODD BYTE
1744 006222 105024      CLRB     (R4)+  ;*TEST INSTRUCTION
1745 006224 001401      BEQ      1$      ;BRANCH IF GOOD
1746                                     ;MODE 2, ODD BYTE FAILED
1747 006226 104001      ERROR    +1      ;CPU ERROR
1748 006230 005304      1$:      DEC      R4      ;POINT TO ODD BYTE
1749 006232 005304      DEC      R4
1750 006234 105224      INCB     (R4)+  ;400=1 0
1751 006236 105124      COMB     (R4)+  ;*TEST INSTRUCTION
1752 006240 100003      BPL      2$      ;BRANCH IF MODE 2 FAILED
    
```

BASE INSTRUCTION SET TESTS

```

1753 006242 005304      DEC      R4      ;POINT TO ODD BYTE
1754 006244 105224      INCB    (R4)+
1755 006246 001401      BEQ     3$
1756                      ;BRANCH IF GOOD
1757 006250 104001      2$:     ERROR   +1      ;MODE 2,ODD BYTE FAILED
1758 006252          3$:
1759                      ;
1760                      ;MSPI:
1763 006252          ;
1764                      ;
1765                      ; TEST CLR, COM, INC - MODE 3
1766 006252 005004      CLR     R4      ;
1767 006254 005014      CLR     (R4)    ;0=0
1768 006256 105114      COMB    (R4)    ;
1769 006260 005214      INC     (R4)    ;0=400
1770 006262 005034      CLR     @R4)+   ;*TEST INSTRUCTION
1771 006264 001401      BEQ     1$      ;BRANCH IF GOOD
1772                      ;MODE 3 FAILED, 400 SHOULD=0
1773 006266 104001      1$:     ERROR   +1      ;CPU ERROR
1774 006270 005304      DEC     R4
1775 006272 005304      DEC     R4      ;R4=0
1776 006274 005134      COM     @R4)+   ;*TEST INSTRUCTION
1777 006276 100004      BPL     2$      ;BRANCH IF BAD
1778 006300 005304      DEC     R4
1779 006302 005304      DEC     R4      ;REPOSITION POINTER
1780 006304 005234      INC     @R4)+   ;*TEST INSTRUCTION
1781 006306 001401      BEQ     3$      ;BRANCH IF GOOD
1782                      ;MODE 3 FAILED
1783 006310 104001      2$:     ERROR   +1      ;CPU ERROR
1784 006312          3$:
1785                      ;
1786                      ;MSPJ:
1789 006312          ;
1790                      ;
1791                      ; TEST CLRB, COMB, INCB - MODE 3, EVEN/ODD BYTE
1792 006312 005004      CLR     R4      ;R4=0
1793 006314 005001      CLR     R1
1794 006316 105101      COMB    R1
1795 006320 005201      INC     R1      ;R1=400
1796 006322 005011      CLR     (R1)
1797 006324 005121      COM     (R1)+   ;400=-1
1798 006326 005011      CLR     (R1)
1799 006330 105111      COMB    (R1)    ;402=000 377
1800 006332 005014      CLR     (R4)
1801 006334 105114      COMB    (R4)
1802 006336 005214      INC     (R4)    ;0=400
1803 006340 105034      CLRB   @R4)+   ;*TEST INSTRUCTION 400=377 000
1804 006342 001401      BEQ     1$      ;BRANCH IF MODE 3 EVEN BYTE CLEARED
1805                      ; TEST INSTRUCTION FAILED
1806 006344 104001      1$:     ERROR   +1      ;CPU ERROR
1807 006346 005304      DEC     R4      ;REPOSITION POINTER
1808 006350 005304      DEC     R4
1809 006352 105134      COMB    @R4)+   ;*TEST INSTRUCTION
1810 006354 005304      DEC     R4
1811 006356 005304      DEC     R4      ;REPOSITION POINTER
1812 006360 105234      INCB   @R4)+   ;*TEST INSTRUCTION
1813 006362 001401      BEQ     3$      ;BRANCH IF GOOD

```

BASE INSTRUCTION SET TESTS

```

1814
1815 006364 104001          2$:  ERROR  +1          ;MODE 3, EVEN BYTE FAILED
1816 006366 005304          3$:  DEC    R4          ;CPU ERROR
1817 006370 005304          DEC    R4
1818 006372 005214          INC    (R4)          ;R4=401
1819 006374 105234          INCB  8(R4)+        ;*TEST INSTRUCTION
1820 006376 001004          BNE   4$            ;BRANCH IF 402 NEQ 0
1821 006400 005304          DEC    R4
1822 006402 005304          DEC    R4          ;R4=401
1823 006404 105034          CLRB  8(R4)+        ;401=0
1824 006406 001401          BEQ   5$            ;BRANCH IF GOOD
1825
1826 006410 104001          4$:  ERROR  +1          ;ODD BYTE FAILED
1827 006412 005304          5$:  DEC    R4          ;CPU ERROR
1828 006414 005304          DEC    R4
1829 006416 105134          COMB  8(R4)+        ;R4=401
1830 006420 005304          DEC    R4          ;403=377
1831 006422 005304          DEC    R4
1832 006424 105234          INCB  8(R4)+        ;*TEST INSTRUCTION
1833 006426 001401          BEQ   7$            ;BRANCH IF GOOD
1834
1835 006430 104001          6$:  ERROR  +1          ;MODE3 ODD BYTE FAILED.
1836 006432          7$:
1837
1838          ;
1841 006432          MSPL:
1842
1843          ;
1844 006432 005004          ; TEST CLR, COM, DEC - MODE 4
1845 006434 105104          CLR   R4
1846 006436 005204          COMB  R4
1847 006440 005014          INC   R4          ;R4=400
1848 006442 005124          CLR  (R4)          ;
1849 006444 005014          COM  (R4)+        ;400=-1
1850 006446 005224          CLR  (R4)          ;
1851 006450 005044          INC  (R4)+        ;402=1
1852 006452 001401          CLR  -(R4)        ;*TEST INSTRUCTION
1853          BEQ   1$            ;BRANCH IF GOOD
1854 006454 104001          ;MODE 4 FAILED
1855 006456 005344          1$:  ERROR  +1          ;CPU ERROR
1856 006460 005114          DEC  -(R4)        ;*TEST INSTRUCTION 400=-2
1857 006462 001405          COM  (R4)          ;400=1
1858 006464 100404          BEQ  2$            ;BRANCH IF BAD
1859 006466 005204          BMI  2$            ;BRANCH IF BAD
1860 006470 005204          INC  R4
1861 006472 005344          INC  R4          ;R4=400
1862 006474 001401          DEC  -(R4)        ;*TEST INSTRUCTION
1863          BEQ   3$            ;BRANCH IF GOOD
1864 006476 104001          2$:  ERROR  +1          ;MODE 4 FAILED
1865 006500          3$:  ;CPU ERROR
1866
1867          ;
1870 006500          MSPM:
1871
1872          ;
1873 006500 005004          ; TEST COMB, INCB, CLRB - MODE 4, ODD BYTE
1874 006502 105104          CLR   R4
          COMB  R4          ;

```


BASE INSTRUCTION SET TESTS

```

1875 006504 005204      INC      R4          ;R4=400
1876 006506 005044      CLR      -(R4)       ;376=0
1877 006510 105114      COMB     (R4)
1878 006512 005224      INC      (R4)+      ;376=001 000
1879 006514 005014      CLR      (R4)
1880 006516 005124      COM      (R4)+      ;400=1
1881 006520 005204      INC      R4          ;R4=403
1882 006522 105044      CLRB     -(R4)      ; TEST INST. CLEAR ODD BYTE (401)
1883 006524 001401      BEQ      2$
1884
1885 006526 104001      1$:      ERROR      +1
1886 006530 005204      2$:      INC      R4
1887 006532 005204      INC      R4          ;R4=403
1888 006534 105144      COMB     -(R4)      ; TEST INST. 401=377
1889 006536 005304      DEC      R4
1890 006540 005304      DEC      R4
1891 006542 105244      INCB     -(R4)      ; TEST INST. 401=0
1892 006544 001401      BEQ      4$
1893
1894 006546 104001      3$:      ERROR      +1
1895 006550 105344      4$:      DECB     -(R4)      ;CPU ERROR
1896 006552 001401      BEQ      6$
1897
1898 006554 104001      5$:      ERROR      +1
1899 006556      6$:
1900
1901      ;
1904 006556      MSPN:
1905
1906      ;
1907 006556 005004      ; TEST CLR, COM, INC - MODE 5
1908 006560 005014      CLR      R4
1909 006562 105114      CLR      (R4)
1910 006564 005224      COMB     (R4)
1911 006566 005054      INC      (R4)+      ;0=400
1912 006570 001401      CLR      0-(R4)     ;*TEST INST. 400=0
1913
1914 006572 104001      BEQ      1$
1915 006574 005204      1$:      ERROR      +1
1916 006576 005204      INC      R4
1917 006600 005154      INC      R4          ;RESET POINTER TO 0
1918 006602 001407      COM      0-(R4)     ;*TEST INST. 376=1
1919 006604 005204      BEQ      2$
1920 006606 005204      INC      R4          ;BRANCH IF BAD
1921 006610 005354      INC      R4
1922 006612 001403      DEC      0-(R4)     ;REPOSITION POINTER
1923 006614 005224      BEQ      2$
1924 006616 105254      INC      (R4)+      ;*TEST INST. 376=0
1925 006620 001401      INCB     0-(R4)     ;BRANCH IF BAD
1926
1927 006622 104001      2$:      ERROR      +1
1928 006624      3$:
1929
1930      ;
1933 006624      MSPO:
1934
1935      ; TEST NEG MODE 5

```

BASE INSTRUCTION SET TESTS

```

1936 005624 005004      CLR      R4
1937 006626 105104      COMB     R4
1938 006630 005204      INC      R4           ;R4=400
1939 006632 005001      CLR      R1
1940 006634 105101      COMB     R1
1941 006636 005301      DEC      R1           ;R1=376
1942 006640 005002      CLR      R2           ;R2=0
1943 006642 005012      CLR      (R2)         ;0=0
1944 006644 005014      CLR      (R4)
1945 006646 005114      COM      (R4)         ;400=-1
1946 006650 005011      CLR      (R1)         ;376=0
1947 006652 005454      NEG      @-(R4)       ;0=0
1948 006654 001401      BEQ      2$           ;BRANCH IF GOOD
1949                                ;NEG FAILED
1950 006656 104001      1$:      ERROR      +1       ;CPU ERROR
1951 006660 005334      2$:      DEC      @-(R4)+    ;0=-1
1952 006662 005454      NEG      @-(R4)       ;0=1
1953 006664 001403      BEQ      3$           ;BRANCH IF BAD
1954 006666 102402      BVS      3$           ;BRANCH IF BAD
1955 006670 100401      BMI      3$           ;BRANCH IF BAD
1956 006672 103401      BCS      4$           ;BRANCH IF GOOD
1957                                ;NEG FAILED
1958 006674 104001      3$:      ERROR      +1       ;CPU ERROR
1959 006676 005334      4$:      DEC      @-(R4)+    ; TEST RESULT OF NEGATE
1960 006700 001401      BEQ      6$           ;BRANCH IF GOOD
1961                                ;RESULT OF NEGATE BAD
1962 006702 104001      5$:      ERROR      +1       ;CPU ERROR
1963 006704 105212      6$:      INCB     (R2)         ;0=1
1964 006706 005454      NEG      @-(R4)       ;0=-1
1965 006710 001403      BEQ      7$
1966 006712 102402      BVS      7$
1967 006714 103001      BCC      7$
1968 006716 100401      BMI      8$           ;BRANCH IF GOOD
1969                                ;BAD NEGATE
1970 006720 104001      7$:      ERROR      +1       ;CPU ERROR
1971 006722 105212      8$:      INCB     (R2)         ;0=0
1972 006724 001401      BEQ      10$          ;BRANCH IF GOOD
1973 006726 104001      9$:      ERROR      +1       ;CPU ERROR
1974 006730      10$:
1975
1976
1978
1980 006730      MSPP:
1981
1982 ;      TEST CLR, COM, INC - MODE 6
1983 006730 005004      CLR      R4
1984 006732 005204      INC      R4
1985 006734 005204      INC      R4           ;R4=2
1986 006736 005001      CLR      R1
1987 006740 105101      COMB     R1
1988 006742 005201      INC      R1           ;R1=400
1989 006744 005011      CLR      (R1)
1990 006746 005121      COM      (R1)+        ;400=-1
1991 006750 005011      CLR      (R1)         ;R1=402
1992 006752 005211      INC      (R1)         ;402=1
1993 006754 005002      CLR      R2           ;R2=0
1994 006756 005012      CLR      (R2)         ;0=0

```

BASE INSTRUCTION SET TESTS

```

1995 006760 005064 000376          CLR      376(R4)          ;400=0
1996 006764 001401                   BEQ      2$              ;BRANCH IF GOOD
1997 006766 104001                   1$:      ERROR      +1          ;CPU ERROR
1998 006770 005364 000376          2$:      DEC      376(R4)        ;400=-1
1999 006774 005164 000400          COM      400(R4)        ;402=-1
2000 007000 001405                   BEQ      3$              ;BRANCH IF BAD
2001 007002 005264 000400          INC      400(R4)        ;402=-2
2002 007006 005264 000400          INC      400(R4)
2003 007012 001401                   BEQ      4$              ;BRANCH IF GOOD
2004                                     ;MODE 6 FAILED
2005 007014 104001                   3$:      ERROR      +1          ;CPU ERROR
2006 007016 005261 177776          4$:      INC      -2(R1)        ;400=0
2007 007022 001401                   BEQ      6$              ;BRANCH IF GOOD
2008                                     ;INC MODE 6 FAILED
2009 007024 104001                   5$:      ERROR      +1          ;CPU ERROR
2010 007026          6$:
2011                                     ;
2012                                     ;
2014                                     ;
2015                                     ;
2017 007026          MSPQ:
2018                                     ;
2019                                     ;      TEST NEG MODE 6
2020 007026 005001          CLR      R1              ;R1=0
2021 007030 005004          CLR      R4
2022 007032 105104          COMB     R4
2023 007034 005204          INC      R4              ;R4=400
2024 007036 005014          CLR      (R4)            ;
2025 007040 005114          COM      (R4)            ;400=-1
2026 007042 005044          CLR      -(R4)          ;376=0
2027 007044 005044          CLR      -(R4)
2028 007046 005224          INC      (R4)+          ;374=1 R4=376
2029 007050 005464 000002          NEG      2(R4)          ;400=1
2030 007054 001403          BEQ      1$              ;NEGATE FAILED
2031 007056 102402          BVS      1$
2032 007060 100401          BMI      1$
2033 007062 103401          BCS      2$              ;BRANCH IF GOOD
2034                                     ;NEGATE FAILED
2035 007064 104001                   1$:      ERROR      +1          ;CPU ERROR
2036 007066 005364 000002          2$:      DEC      2(R4)        ; TEST RESULT OF NEGATE
2037 007072 001401                   BEQ      4$              ;BRANCH IF GOOD
2038                                     ;RESULT OF NEGATE FAILED
2039 007074 104001                   3$:      ERROR      +1          ;CPU ERROR
2040 007076 005464 000000          4$:      NEG      0(R4)        ;+0=0
2041 007102 001401                   BEQ      5$              ; BRANCH IF GOOD
2042                                     ;NEGATE FAILED
2043 007104 104001                   ERROR      +1          ;CPU ERROR
2044 007106 105461 000374          5$:      NEGB     374(R1)        ;374=0 377
2045 007112 102403          BVS      6$
2046 007114 001402          BEQ      6$
2047 007116 100001          BPL      6$
2048 007120 103401          BCS      7$              ;BRANCH IF GOOD
2049                                     ;NEGATE FAILED
2050 007122 104001                   6$:      ERROR      +1          ;CPU ERROR
2051 007124 105261 000374          7$:      INCB     374(R1)        ;374=0
2052 007130 001401                   BEQ      9$              ;BRANCH IF GOOD
2053                                     ;NEGATE FAILED

```


BASE INSTRUCTION SET TESTS

```

2115 007270      4$:
2116
2117
2120 007270      ;
                MSPT:
2121
2122      ;      TEST SINGLE OPERAND MODE 2 REG 7
2123 007270 005004 CLR      R4
2124 007272 105104 COMB    R4
2125 007274 005204 INC      R4
2126 007276 005027 CLR      (R7)+      ;R4=400
2127 007300      1$: .WORD  -1      ;CLEAR NEXT LOCATION
2128 007302 001401 BEQ      3$      ;SETUP INITIAL DATA
2129 007304 104001 ERROR   +1      ;BRANCH IF GOOD
2130 007306      2$:
                3$:      ;CPU ERROR
2131
2132
2134
2136 007306      MSPU:
2137
2138      ;      TEST TST MODE 0
2139 007306 005004 CLR      R4      ;R4=0
2140 007310 000277 SCC      ;CONDITION CODES =1111
2141 007312 000244 CLZ      ;CC=1011
2142 007314 005704 TST      R4      ;*TEST INSTRUCTION
2143 007316 103403 BCS      1$
2144 007320 102402 BVS      1$
2145 007322 100401 BMI      1$      ;BRANCH IF ERROR
2146 007324 001401 BEQ      2$      ;BRANCH IF GOOD
2147 007326 104001 ERROR   +1      ;CPU ERROR
2148
2149 007330 005304      2$: DEC      R4      ;TST MODE 0 FAILED
2150 007332 000277 SCC
2151 007334 000250 CLN      ;R4=-1
2152 007336 005704 TST      R4      ;CC=0111
2153 007340 103403 BCS      3$      ;*TEST INSTRUCTION MODE 0
2154 007342 102402 BVS      3$      ;BRANCH IF ERROR
2155 007344 001401 BEQ      3$
2156 007346 100401 BMI      4$
2157 007350 104001 ERROR   +1      ;BRANCH IF GOOD
2158
2159 007352      3$:      ;CPU ERROR
                4$:      ;TST FAILED
2160
2161
2164 007352      ;
                MSPVO:
2165
2166      ;      TEST TST MODE 0 BYTE
2167 007352 005004 CLR      R4
2168 007354 105104 COMB    R4      ;0=000 377
2169 007356 000277 SCC
2170 007360 000250 CLN      ;CC=0111
2171 007362 105704 TSTB   R4      ;*TEST INSTRUCTION ON EVEN BYTE
2172 007364 102403 BVS      1$      ;BRANCH IF ERROR
2173 007366 103402 BCS      1$
2174 007370 102401 BVS      1$
2175 007372 100401 BMI      2$
2176 007374 104001 ERROR   +1      ;BRANCH IF GOOD
2177
                ;CPU ERROR
                ;

```


BASE INSTRUCTION SET TESTS

```

2178 007376 005204      2$:  INC    R4      ;POINT TO 1
2179 007400 105704      TSTB   R4      ; TEST INSTRUCTION
2180 007402 001401      BEQ    4$      ;BRANCH IF GOOD
2181 007404 104001      3$:  ERROR  +1   ;CPU ERROR
2182                               ;TST FAILED ON BYTE
2183 007406      4$:
2184
2185      ;
2188 007406      ;MSPV:
2189
2190      ; TEST TST MODE 1
2191 007406 005004      CLR    R4
2192 007410 005014      CLR    (R4)    ;0=0
2193 007412 000277      SCC
2194 007414 000244      CLZ      ;CC=1011
2195 007416 005714      TST    (R4)    ;*TEST INSTRUCTION IN MODE 1
2196 007420 103403      BCS    1$      ;BRANCH IF ERROR
2197 007422 102402      BVS    1$
2198 007424 100401      BMI    1$      ;
2199 007426 001401      BEQ    2$      ;BRANCH IF GOOD
2200 007430 104001      1$:  ERROR  +1   ;CPU ERROR
2201
2202 007432 005214      2$:  INC    (R4)    ;0=1
2203 007434 000277      SCC
2204 007436 005714      TST    (R4)    ; TEST INSTRUCTION
2205 007440 001403      BEQ    3$      ;BRANCH IF ERROR
2206 007442 102402      BVS    3$
2207 007444 103401      BCS    3$
2208 007446 100001      BPL    4$      ;BRANCH IF GOOD
2209 007450 104001      3$:  ERROR  +1   ;CPU ERROR
2210                               ;TST FAILED MODE 1
2211 007452      4$:
2212
2213      ;
2216 007452      ;MSPX:
2217
2218      ; TEST TST MODE 1 BYTE
2219 007452 005004      CLR    R4      ;R4=0
2220 007454 005014      CLR    (R4)
2221 007456 105114      COMB   (R4)
2222 007460 005214      INC    (R4)    ;0=001 000
2223 007462 000277      SCC
2224 007464 000244      CLZ      ;CC=1011
2225 007466 105714      TSTB   (R4)    ;*TEST INSTRUCTION
2226 007470 103403      BCS    1$      ;BRANCH IF ERROR
2227 007472 102402      BVS    1$      ;
2228 007474 100401      BMI    1$
2229 007476 001401      BEQ    2$      ;BRANCH IF GOOD
2230 007500 104001      1$:  ERROR  +1   ;CPU ERROR
2231
2232 007502 005204      2$:  INC    R4      ;R4=1
2233 007504 000277      SCC
2234 007506 105714      TSTB   (R4)    ; TEST INSTRUCTION
2235 007510 001403      BEQ    3$      ;BRANCH IF ERROR
2236 007512 100402      BMI    3$
2237 007514 102401      BVS    3$
2238 007516 103001      BCC    4$      ;BRANCH IF GOOD

```

BASE INSTRUCTION SET TESTS

```

2239 007520 104001      3$:  ERROR  +1          ;CPU ERROR
2240                                     ;
2241 007522      4$:
2242
2243      ;
2246 007522      MSPY:
2247
2248      ;
2249 007522 005004      ; TEST TST MODE 2
2250 007524 005024      CLR R4          ;
2251 007526 005014      CLR (R4)+       ;0=0
2252 007530 005114      CLR (R4)
2253 007532 005004      COM (R4)          ;2=-1
2254 007534 000277      CLR R4          ;R4=0
2255 007536 000244      SCC
2256 007540 005724      CLZ          ;CC=1011
2257 007542 103403      TST (R4)+       ; TEST INSTRUCTION
2258 007544 102402      BCS 1$          ;BRANCH IF ERROR
2259 007546 100401      BVS 1$
2260 007550 001401      BMI 1$
2261 007552 104001      BEQ 2$          ;BRANCH IF GOOD
2262      1$:  ERROR  +1          ;CPU ERROR
2263 007554 005724      2$:  TST (R4)+       ;MODE 2 TEST FAILED
2264 007556 103403      BCS 3$          ;TST LOC2
2265 007560 102402      BVS 3$
2266 007562 001401      BEQ 3$
2267 007564 100401      BMI 4$
2268 007566 104001      3$:  ERROR  +1          ;CPU ERROR
2269      ;MODE 2 FAILED
2270 007570      4$:
2271
2272      ;
2275 007570      MSPZ:
2276
2277      ;
2278 007570 005004      ; TEST TST MODE 2 BYTE
2279 007572 005024      CLR R4
2280 007574 105144      CLR (R4)+       ;
2281 007576 005304      COMB -(R4)      ;0=377 000
2282 007600 000277      DEC R4          ;R4=0
2283 007602 000244      SCC
2284 007604 105724      CLZ          ;CC=1011
2285 007606 102403      TSTB (R4)+      ;
2286 007610 103402      BVS 1$          ;BRANCH IF ERROR
2287 007612 100401      BCS 1$
2288 007614 001401      BMI 1$
2289 007616 104001      BEQ 2$          ;BRANCH IF GOOD
2290      1$:  ERROR  +1          ;CPU ERROR
2291 007620 000277      2$:  SCC
2292 007622 000250      CLN          ;CC=0111
2293 007624 105724      TSTB (R4)+      ;
2294 007626 001403      BEQ 3$          ;
2295 007630 103402      BCS 3$
2296 007632 102401      BVS 3$
2297 007634 100401      BMI 4$
2298 007636 104001      3$:  ERROR  +1          ;BRANCH IF GOOD
2299      ;CPU ERROR
          ;MODE 2 ODD BYTE FAILED

```

BASE INSTRUCTION SET TESTS

```

2300 007640      4:
2301
2302            ;
2305 007640      MSPAA:
2306
2307            ;      TEST TST MODE 3
2308 007640 005004      CLR      R4
2309 007642 005014      CLR      (R4)
2310 007644 105114      COMB     (R4)
2311 007646 005214      INC      (R4)
2312 007650 005034      CLR      @ (R4)+
2313 007652 005004      CLR      R4
2314 007654 000277      SCC
2315 007656 000244      CLZ
2316 007660 005734      TST      @ (R4)+
2317 007662 103403      BCS      1:
2318 007664 102402      BVS      1:
2319 007666 100401      BMI      1:
2320 007670 001401      BEQ      2:
2321 007672 104001      1:      ERROR    +1
2322
2323 007674 005304      2:      DEC      R4
2324 007676 005304      DEC      R4
2325 007700 005334      DEC      @ (R4)+
2326 007702 005004      CLR      R4
2327 007704 000277      SCC
2328 007706 000250      CLN
2329 007710 005734      TST      @ (R4)+
2330 007712 103403      BCS      3:
2331 007714 001402      BEQ      3:
2332 007716 102401      BVS      3:
2333 007720 100401      BMI      4:
2334
2335 007722 104001      3:      ERROR    +1
2336 007724      4:
2337
2338            ;
2341 007724      MSPBB:
2342
2343            ;      TEST TST MODE 3 AUTO-INC
2344 007724 005004      CLR      R4
2345 007726 005014      CLR      (R4)
2346 007730 105114      COMB     (R4)
2347 007732 005214      INC      (R4)
2348 007734 005001      CLR      R1
2349 007736 105101      COMB     R1
2350 007740 005201      INC      R1
2351 007742 005011      CLR      (R1)
2352 007744 000277      SCC
2353 007746 005734      TST      @ (R4)+
2354 007750 103403      BCS      1:
2355 007752 102402      BVS      1:
2356 007754 100401      BMI      1:
2357 007756 001401      BEQ      2:
2358 007760 104001      1:      ERROR    +1
2359
2360 007762 005304      2:      DEC      R4

```

;CC SHOULD = 0100

BASE INSTRUCTION SET TESTS

```

2361 007764 005304      DEC      R4
2362 007766 005704      TST      R4          ;SEE IF AUTO-INC WORKED
2363 007770 001401      BEQ      4#          ;ERROR IF R4 NE 0
2364 007772 104001      3#:      ERROR     +1          ;CPU ERROR
2365                                     ;AUTO-INC FIALED
2366 007774      4#:
2367
2368
2371 007774      ;MSTB3:
2372
2373      ;      TEST TST MODE 3 BYTE
2374 007774 005004      CLR      R4
2375 007776 005014      CLR      (R4)
2376 010000 105114      COMB     (R4)
2377 010002 005214      INC      (R4)
2378 010004 005214      INC      (R4)          ;0=401
2379 010006 005001      CLR      R1
2380 010010 105101      COMB     R1
2381 010012 005201      INC      R1          ;R1=400
2382 010014 005011      CLR      (R1)
2383 010016 005111      COM      (R1)
2384 010020 105011      CLR      (R1)          ;400=377 000
2385 010022 105734      TSTB     8(R4)+      ;** TEST INSTRUCTION
2386 010024 001403      BEQ      1#          ;ERROR IF EQUAL
2387 010026 103402      BCS      1#          ;ERROR IF CARRY SET
2388 010030 102401      BVS      1#          ;ERROR IR OVERFLOW
2389 010032 100401      BMI      2#          ;BRANCH IF MINUS
2390 010034 104001      1#:      ERROR     +1          ;CPU ERROR
2391                                     ;CC ERROR
2392 010036 005304      2#:      DEC      R4
2393 010040 005304      DEC      R4
2394 010042 001401      BEQ      4#          ;BRANCH IF AUTO-INC WORKED
2395 010044 104001      3#:      ERROR     +1          ;CPU ERROR
2396                                     ;AUTO-INC FAILED
2397 010046      4#:
2398
2399      ;MST4:
2402 010046
2403
2404      ;      TEST TST MODE 4
2405 010046 005004      CLR      R4
2406 010050 005014      CLR      (R4)          ;0=0
2407 010052 005204      INC      R4
2408 010054 005204      INC      R4          ;R4=2
2409 010056 000277      SCC
2410 010060 000244      CLZ          ;CC=1011
2411 010062 005744      TST      -(R4)      ;**TEST INTRUCTION
2412 010064 103403      BCS      1#          ;ERROR IF CARRY
2413 010066 102402      BVS      1#          ;ERROR IF OVERFLOW
2414 010070 100401      BMI      1#          ;ERROR IF MINUS
2415 010072 001401      BEQ      2#          ;BRANCH IF GOOD
2416 010074 104001      1#:      ERROR     +1          ;CPU ERROR
2417                                     ;CC WRONG
2418 010076 005704      2#:      TST      R4
2419 010100 001401      BEQ      4#          ;INSURE CORRECT AUTO DEC
2420                                     ;BRANCH IF GOOD AUTO-DEC
2421 010102 104001      3#:      ERROR     +1          ;BAD AUTO-DEC
                                     ;CPU ERROR

```

BASE INSTRUCTION SET TESTS

```

2422 010104      4$:
2423
2424      ;
2427 010104      ;MST4B:
2428
2429      ;      TEST TST MODE 4 BYTE
2430 010104 005004      CLR      R4
2431 010106 005014      CLR      (R4)
2432 010110 005114      COM      (R4)
2433 010112 105114      COMB     (R4)          ;0=377 000
2434 010114 000277      SCC
2435 010116 005204      INC      R4
2436 010120 005204      INC      R4          ;R4=2
2437 010122 105744      TSTB     (R4)        ;**TEST INSTRUCTION
2438 010124 001403      BEQ      1$          ;ERROR IF EQUAL TO 0
2439 010126 103402      BCS      1$          ;ERROR IF CARRY
2440 010130 102401      BVS      1$          ;ERROR IF OVERFLOW
2441 010132 100401      BMI      2$          ;BRANCH IF MINUS
2442 010134 104001      1$:      ERROR     +1          ;CPU ERROR
2443      ;CC SHOULD EQUAL 0100
2444 010136 105744      2$:      TSTB     (R4)        ;**TEST EVEN BYTE
2445 010140 001401      BEQ      4$          ;BRANCH IF GOOD
2446 010142 104001      3$:      ERROR     +1          ;CPU ERROR
2447      ;CC SHOULD EQUAL 0100 AND R4= 1
2448 010144      4$:
2449      ;
2452 010144      ;MST5:
2453
2454      ;      TEST TST MODE 5
2455 010144 005004      CLR      R4          ;
2456 010146 005024      CLR      (R4).      ;0=0, R4=2
2457 010150 000277      SCC
2458 010152 000244      CLZ
2459 010154 005754      TST      @-(R4)     ;CC=1011
2460 010156 103403      BCS      1$          ;TEST INSTRUCTION
2461 010160 102402      BVS      1$          ;ERROR IF CARRY
2462 010162 100401      BMI      1$          ;ERROR IF OVERFLOW
2463 010164 001401      BEQ      2$          ;ERROR IF MINUS
2464 010166 104001      1$:      ERROR     +1          ;BRANCH IF GOOD
2465      ;CPU ERROR
2466 010170 005704      2$:      TST      R4          ;CC WRONG, SHOULD = 0100
2467 010172 001401      BEQ      4$          ;BRANCH IF AUTO-DEC WORKED
2468 010174 104001      3$:      ERROR     +1          ;CPU ERROR
2469      ;AUTO-DEC FAILED
2470 010176      4$:
2471
2472      ;
2475 010176      ;MST5B:
2476
2477      ;      TEST TST MODE 5 BYTE
2478 010176 005004      CLR      R4
2479 010200 005014      CLR      (R4)
2480 010202 105114      COMB     (R4)
2481 010204 005214      INC      (R4)          ;0=400
2482 010206 005034      CLR      @-(R4).    ;400=0, R4=2
2483 010210 005154      COM      @ (R4)
2484 010212 105134      COMB     @-(R4).    ;400=377 000 R4=2
    
```

BASE INSTRUCTION SET TESTS

```

2485 010214 105754      TSTB      0 (R4)      ;**TEST INSTRUCTION
2486 010216 103403      BCS       1$         ;ERROR IF CARRY
2487 010220 100402      BMI       1$         ;ERROR IF MINUS
2488 010222 102401      BVS       1$         ;ERROR IF OVERFLOW
2489 010224 001401      BEQ       2$         ;BRANCH IF GOOD
2490 010226 104001      ERROR    +1         ;CPU ERROR
2491                1$:
2492 010230 005224      INC       (R4)+      ;CC SHOULD = 0100
2493 010232 105754      TSTB     0-(R4)     ;0=401
2494 010234 100401      BMI       4$         ;**TEST INSTRUCTION
2495                3$:
2496 010236 104001      ERROR    +1         ;BRANCH IF GOOD
2497 010240                4$:
2498
2499                ;
2502 010240                ;MST6:
2503
2504                ;
2505 010240 005004      CLR       R4         ;TEST TST MODE 6
2506 010242 005014      CLR      (R4)        ;0=0
2507 010244 105104      COMB     R4
2508 010246 005204      INC      R4         ;R4=400
2509 010250 005014      CLP      (R4)
2510 010252 005114      COM      (R4)        ;400=-1
2511 010254 005764 177400 TST      -400(R4)    ;**TEST LOCATION 0
2512 010260 103403      BCS       1$         ;ERROR IF CARRY
2513 010262 102402      BVS       1$         ;ERROR IF OVERFLOW
2514 010264 100401      BMI       1$         ;ERROR IF MINUS
2515 010266 001401      BEQ       2$         ;BRANCH IF ZERO
2516 010270 104001      ERROR    +1         ;CPU ERROR
2517                1$:
2518 010272 005004      CLR       R4         ;CC ARE WRONG
2519 010274 005764 000400 TST      400(R4)     ;
2520 010300 001401      BEQ       3$         ;TST LOCATION 400
2521 010302 100401      BMI       4$         ;ERROR IF EQUAL
2522 010304 104001      ERROR    +1         ;BRANCH IF MINUS
2523                3$:
2524 010306                4$:
2525                ;
2526                ;
2529 010306                ;MST7:
2530
2531                ;
2532 010306 005004      CLR       R4         ;TEST TST MODE 7
2533 010310 005014      CLR      (R4)
2534 010312 005124      COM      (R4)+      ;0=-1
2535 010314 005014      CLR      (R4)        ;2=0
2536 010316 005002      CLR      R2         ;R2=0
2537 010320 005004      CLR      R4
2538 010322 105104      COMB     R4
2539 010324 005204      INC      R4         ;R4=400
2540 010326 005014      CLR      (R4)        ;400=0
2541 010330 005774 177402 TST      0-376(R4)   ;**TEST LOCATION 0
2542 010334 103403      BCS       1$         ;ERROR IF CARRY
2543 010336 102402      BVS       1$         ;ERROR IF OVERFLOW
2544 010340 001401      BEQ       1$         ;ERROR IF ZERO
2545 010342 100401      BMI       2$         ;BRANCH IF GOOD

```


BASE INSTRUCTION SET TESTS

```

2546 010344 104001      1$:  ERROR  +1          ;CPU ERROR
2547                    ;ERROR IN CC
2548 010346 005222      2$:  INC    (R2).        ;0=-2 R2=2
2549 010350 005774 177402  TST    0-376(R4)      ;** CHECK CONTENTS OF LOCATION 2
2550 010354 100401      BMI    3$             ;ERROR IF MINUS
2551 010356 001401      BEQ    4$             ;BRANCH IF GOOD
2552 010360 104001      3$:  ERROR  +1          ;CPU ERROR
2553                    ;CC SHOULD = 0100
2554 010362      4$:
2555
2556
2558
2559
2560
2561
2562
2563      .SBTTL ***** DOUBLE OPERAND TESTS *****
2565 010362      ;MMMO:
2566
2567      ; TEST MOVE MODE 0
2568 010362 005004      CLR    R4             ;R4=0
2569 010364 005001      CLR    R1
2570 010366 005101      COM    R1             ;R1=-1
2571 010370 010104      MOV    R1,R4         ;**TEST MOVE INSTRUCTION
2572 010372 001402      BEQ    1$             ;ERROR IF R4=0
2573 010374 102401      BVS    1$             ;ERROR IF OBERFLOW
2574 010376 100401      BMI    2$             ;BRANCH IF MINUS
2575 010400 104001      1$:  ERROR  +1          ;CPU ERROR
2576                    ;CC SHOULD = 100-, R4 SHOULD=-1
2577 010402 005204      2$:  INC    R4             ;R4 SHOULD =0
2578 010404 001375      BNE    1$             ;ERROR IF R4 NE 0
2579
2580
2583 010406      ;MDAO:
2584
2585      ; TEST ADD MODE 0
2586 010406 005004      CLR    R4             ;R4=0
2587 010410 005001      CLR    R1
2588 010412 005101      COM    R1             ;R1=-1
2589 010414 060104      ADD    R1,R4         ;** TEST ADD OF R1 TO R4
2590 010416 001403      BEQ    1$             ;ERROR IF ZERO
2591 010420 103402      BCS    1$             ;ERROR IF CARRY
2592 010422 102401      BVS    1$             ;ERROR IF OVERFLOW
2593 010424 100401      BMI    2$             ;BRANCH IF MINUS
2594 010426 104001      1$:  ERROR  +1          ;CPU ERROR
2595                    ;CC SHOULD = 1000 , R4=-1
2596 010430 005204      2$:  INC    R4             ;R4 SHOULD =0
2597 010432 001401      BEQ    4$             ;BRANCH IF R4=0
2598 010434 104001      3$:  ERROR  +1          ;CPU ERROR
2599                    ;R4 SHOULD = 0
2600 010436      4$:
2601
2602
2605 010436      ;MDSO:
2606
2607      ; TEST SUB MODE 0
2608 010436 005004      CLR    R4

```

***** DOUBLE OPERAND TESTS *****

```

2609 010440 005001          CLR    R1
2610 010442 005201          INC    R1                ;R1=1 R4=0
2611 010444 160104          SUB    R1,R4            ;**TEST OF R4-R1, R4=-1
2612 010446 102403          BVS   1$                ;ERROR IF V SET
2613 010450 103002          BCC   1$                ;ERROR IF NO CARRY
2614 010452 001401          BEQ   1$                ;ERROR IF =0
2615 010454 100401          BMI   2$                ;BRANCH IF MINUS
2616 010456 104001          1$:  ERROR    +1        ;CPU ERROR
2617                                ;CC SHOULD = 1001
2618 010460 005101          2$:  COM    R1                ;R1=1
2619 010462 005201          INC    R1                ;GET TWO'S COMPLIMENT, R1=-1
2620 010464 160104          SUB    R1,R4            ;**TEST R4-R1 (1- 1= 0
2621 010466 001401          BEQ   4$                ;BRANCH IF ZERO
2622 010470 104001          3$:  ERROR    +1        ;CPU ERROR
2623                                ;CC SHOULD = 0100
2624 010472          4$:
2625
2626
2629 010472          ; MDM27:
2630
2631          ; TEST MOV MODE 27,00
2632 010472 000257          CCC
2633 010474 012704 125252          MOV    #125252,R4        ;CC=0000
2634 010500 001401          BEQ   1$                ;**TEST MOVE
2635 010502 100401          BMI   2$                ;ERROR IF = 0
2636 010504 104001          1$:  ERROR    +1        ;BRANCH IF MINUS
2637                                ;CPU ERROR
2638 010506 012701 052525          2$:  MOV    #052525,R1    ;CC SHOULD = 1000
2639 010512 100401          BMI   3$                ;**TEST MOVE
2640 010514 001001          BNE   4$                ;ERROR IF MINUS
2641 010516 104001          3$:  ERROR    +1        ;BRANCH IF NE 0
2642                                ;CPU ERROR
2643 010520 060104          4$:  ADD    R1,R4        ;CC SHOULD = 0000
2644 010522 100401          BMI   6$                ;R1+R4=-1
2645 010524 104001          5$:  ERROR    +1        ;BRANCH IF MINUS
2646                                ;CPU ERROR
2647 010526 005204          6$:  INC    R4                ;MOV FAILED
2648 010530 001375          BNE   5$                ;R4+1=0
2649                                ;ERROR IF NOT ZERO
2650
2653 010532          ; MBI00:
2654
2655          ; TEST BIC, BIS MODE 0,0
2656 010532 005004          CLR    R4
2657 010534 005104          COM    R4                ;R4=-1
2658 010536 012701 125252          MOV    #125252,R1        ;SETUP R1 TEST DATA
2659 010542 012702 052525          MOV    #052525,R2        ;R2=COMPLIMENT OF R1
2660 010546 000261          SEC
2661 010550 040104          BIC    R1,R4            ;**TEST BIC WITH CARRY SET
2662 010552 103003          BCC   1$                ;ERROR IF NO CARRY
2663 010554 102402          BVS   1$                ;ERROR IF OVERFLOW
2664 010556 001401          BEQ   1$                ;ERROR IF 0
2665 010560 100001          BPL   2$                ;BRANCH IF PLUS
2666 010562 104001          1$:  ERROR    +1        ;CPU ERROR
2667                                ;CC SHOULD = 0001
2668 010564 020402          2$:  CMP    R4,R2        ;COMPARE CONTENTS OF R4 AND R2
2669 010566 001401          BEQ   4$                ;BRANCH IF EQUAL

```

***** DOUBLE OPERAND TESTS *****

```

2670 010570 104001      3$:  ERROR  +1      ;CPU ERROR
2671                                     ;R4 AND R2 SHOULD BE EQUAL
2672 010572 005301      4$:  DEC    R1      ;R1=125251
2673 010574 050201      BIS    R2,R1      ;BIS 052525 AND 125251=177775
2674 010576 100401      BMI    6$      ;BRANCH IF MIUS VALUE
2675 010600 104001      5$:  ERROR  +1      ;CPU ERROR
2676                                     ;BAD BIS OPERATION
2677 010602 005201      6$:  INC    R1
2678 010604 005201      INC    R1
2679 010606 005201      INC    R1      ;R1=0
2680 010610 001373      BNE    5$      ;ERROR IF NE 0
2681
2682
2685 010612      ;
2686      ;MBC00:
2687      ;
2688 010612 012701 125252      ;
2689 010616 012704 100000      MOV    #125252,R1      ;R1=125252
2690 010622 012702 052525      MOV    #100000,R4      ;R4=100000
2691 010626 030401      MOV    #052525,R2      ;F=052525
2692 010630 001401      BIT    R4,R1      ;**TEST OF BIT .CC=1000
2693 010632 100401      BEQ    1$      ;ERROR IF EQ 0
2694 010634 104001      BMI    2$      ;BRANCH IF GOOD
2695      1$:  ERROR  +1      ;CPU ERROR
2696 010636 020401      2$:  CMP    R4,R1      ;CC SHOULD = 1000
2697 010640 001402      BEQ    3$      ;*TEST 100000 125252=25252
2698 010642 103001      BCC    3$      ;ERROR IF EQUAL 0
2699 010644 100401      BMI    4$      ;ERROR IF CARRY CLEARED
2700 010646 104001      3$:  ERROR  +1      ;BRANCH IF GOOD
2701                                     ;CPU ERROR
2702 010650 020104      4$:  CMP    R1,R4      ;CC SHOULD = 0010
2703 010652 001403      BEQ    5$      ;125252-100000 = 25252
2704 010654 103402      BCS    5$      ;ERROR IF EQUAL
2705 010656 102401      BVS    5$      ;ERROR IF CARRY
2706 010660 100001      BPL    6$      ;ERROR IF OVERFLOW
2707 010662 104001      5$:  ERROR  +1      ;BRANCH IF GOOD
2708                                     ;CPU ERROR
2709 010664 005004      6$:  CLR    R4      ;CC SHOULD =0001
2710 010666 005204      INC    R4      ;R4=1
2711 010670 000277      SCC
2712 010672 030401      BIT    R4,R1      ;R4 * R1 = 2
2713 010674 001401      BEQ    8$      ;BRANCH IF GOOD
2714 010676 104001      7$:  ERROR  +1      ;CPU ERROR
2715                                     ;CC SHOULD = 0101
2716 010700      8$:
2717
2718      ;
2721 010700      ;MM11:
2722
2723      ;
2724 010700 012704 000400      ;
2725 010704 012701 000402      TEST MOV, MOVB MODE 1.1 AND SIGN EXT ON MOVB TO GPR
2726 010710 005014      MOV    #400,R4      ;R4=400
2727 010712 005114      MOV    #402,R1      ;R1=402
2728 010714 005011      CLR    (R4)      ;
2729 010716 105111      COM    (R4)      ;400=-1
2730 010720 005002      CLR    (R1)      ;
2731                                     ;402=000 377
2732                                     ;R2=0

```

***** DOUBLE OPERAND TESTS *****

```

2731 010722 012703 000405      MOV      #405,R3          ;R3=405
2732 010726 000277              SCC                      ;CC=1111
2733 010730 011412      MOV      (R4),(R2)       ;MOV 400 TO 0 ,0= 1
2734 010732 001403      BEQ      1$              ;ERROR IF 0
2735 010734 102402      BVS      1$              ;ERROR IF OVERFLOW
2736 010736 103001      BCC      1$              ;ERROR IF NO CARRY
2737 010740 100401      BMI      2$              ;BRANCH IF GOOD
2738 010742 104001      1$:      ERROR      +1          ;CPU ERROR
2739                                ;CC SHOULD =1001
2740 010744 005212      2$:      INC      (R2)          ;0=0
2741 010746 001004      BNE      3$              ;ERROR IF NOT 0
2742 010750 000257      CCC                      ;CC=0000
2743 010752 111113      MOVVB   (R1),(R3)       ;405=377
2744 010754 001401      BEQ      3$              ;ERROR IF EQUAL
2745 010756 100401      BMI      4$              ;BRANCH IF GOOD
2746 010760 104001      3$:      ERROR      +1          ;CPU ERROR
2747
2748 010762 105213      4$:      INCB   (R3)          ;405=0
2749 010764 001375      BNE      3$              ;ERROR IF 405 NOT 0
2750                                ;CHECK THAT SIGN EXTENSION OCCURS ON A MOVB TO GENERAL REGISTER.
2751 010766 005002      CLR      R2              ;INIT R2 TO ZERO.
2752 010770 111102      MOVVB   (R1),R2         ;MOVE 377 TO R2
2753 010772 100005      BPL      5$              ;ERROR! BIT 15 SHOULD BE SET.
2754 010774 102404      BVS      5$              ;V BIT SHOULD BE CLEARED
2755 010776 103403      BCS      5$              ;CARRY BIT SHOULD BE UNAFFECTED
2756 011000 022702 177777      CMP      #177777,R2     ;TEST R2
2757 011004 001401      BEQ      6$              ;SIGN EXTENDED THROUGH UPPER BYTE
2758                                ;ERROR! BYTE SHOULD HAVE
2759                                ;SIGN EXTENDED THROUGH UPPER BYTE
2760 011006 104001      5$:      ERROR      +1          ;CPU ERROR
2761 011010      6$:
2762
2763                                ;
2766 011010      MA11:
2767
2768                                ; TEST ADD MODE 1,1
2769 011010 012704 000400      MOV      #400,R4        ;R4=400
2770 011014 012701 000402      MOV      #402,R1        ;R1=402
2771 011020 012714 177753      MOV      #-25,(R4)      ;400=-25
2772 011024 012711 000024      MOV      #24,(R1)       ;402=24
2773 011030 061114      ADD      (R1),(R4)      ;-25+24=-1
2774 011032 001404      BEQ      1$              ;ERROR IF 0
2775 011034 103403      BCS      1$              ;ERROR IF CARRY
2776 011036 100002      BPL      1$              ;ERROR IF POSITIVE RESULT
2777 011040 005214      INC      (R4)           ;-1+1=0
2778 011042 001401      BEQ      2$              ;BRANCH IF GOOD
2779 011044 104001      1$:      ERROR      +1          ;CPU ERROR
2780                                ;CC SHOULD = 1000
2781 011046      2$:
2782
2783                                ;
2786 011046      MS11:
2787
2788                                ; TEST SUB MODE 1,1
2789 011046 012704 000400      MOV      #400,R4        ;R4=400
2790 011052 012701 000404      MOV      #404,R1        ;R1=404
2791 011056 012714 000003      MOV      #3,(R4)       ;400=3

```

***** DOUBLE OPERAND TESTS *****

```

2792 011062 012711 000006      MOV      #6,(R1)          ;406=6
2793 011066 000277              SCC                      ;CC=1111
2794 011070 161411              SUB      (R4),(R1)       ;5 3-3
2795 011072 001402              BEQ      1$              ;ERROR IF 0
2796 011074 100401              BMI      1$              ;ERROR IF MINUS
2797 011076 103001              BCC      2$              ;BRANCH IF GOOD
2798 011100 104001              1$:  ERROR      +1        ;CPU ERROR
2799                                ;CC SHOULD = 0000
2800 011102 161411              2$:  SUB      (R4),(R1)   ;3-3=0
2801 011104 001375              BNE      1$              ;ERROR IF NOT 0
2802
2803
2806 011106                      ;
2807                                MBB11:
2808                                ;
2809 011106 012704 000400      MOV      #400,R4         ;R4=400
2810 011112 012701 000402      MOV      #402,R1         ;R1=402
2811 011116 012714 052525      MOV      #052525,(R4)    ;400=052525
2812 011122 012711 125252      MOV      #125252,(R1)    ;402=125252
2813 011126 051411              BIS      (R4),(R1)       ;R4 V R1 = -1
2814 011130 001401              BEQ      1$              ;ERROR IF 0
2815 011132 100401              BMI      2$              ;BRANCH IF GOOD
2816 011134 104001              1$:  ERROR      +1        ;CPU ERROR
2817                                ;CC SHOULD = 1000
2818 011136 005211              2$:  INC      (R1)        ;402=0
2819 011140 001401              BEQ      4$              ;BRANCH IF GOOD
2820 011142 104001              3$:  ERROR      +1        ;CPU ERROR
2821                                ;CC SHOULD = 0100
2822 011144 005311              4$:  DEC      (R1)        ;402=-1
2823 011146 041411              BIC      (R4),(R1)       ;R1=125252
2824 011150 001401              BEQ      5$              ;ERROR IF 0
2825 011152 100401              BMI      6$              ;BRANCH IF GOOD
2826 011154 104001              5$:  ERROR      +1        ;CPU ERROR
2827                                ;CC SHOULD = 1000
2828 011156 005111              6$:  COM      (R1)        ;402=052525
2829 011160 041114              BIC      (R1),(R4)       ;400=0
2830 011162 001401              BEQ      8$              ;BRANCH IF GOOD
2831 011164 104001              7$:  ERROR      +1        ;CPU ERROR
2832                                ;CC SHOULD = 0100
2833 011166                      8$:
2834
2835                                ;
2838 011166                      ;
2839                                MBC11:
2840                                ;
2841 011166 012704 000400      MOV      #400,R4         ;R4=400
2842 011172 012714 052525      MOV      #052525,(R4)    ;400=052525
2843 011176 012701 000402      MOV      #402,R1         ;R1=402
2844 011202 012711 125252      MOV      #125252,(R1)    ;402=125252
2845 011206 000241              CLC                      ;CLEAR CARRY
2846 011210 031411              BIT      (R4),(R1)       ;**052525+125252=0
2847 011212 103401              BCS      1$              ;ERROR IF CARRY
2848 011214 001401              BEQ      2$              ;BRANCH IF GOOD
2849 011216 104001              1$:  ERROR      +1        ;CPU ERROR
2850                                ;CC SHOULD = 0100
2851 011220 021411              2$:  CMP      (R4),(R1)   ;400-402=125253
2852 011222 001403              BEQ      3$              ;ERROR IF ZERO

```

***** DOUBLE OPERAND TESTS *****

```

2853 011224 103002      BCC      3$      ;ERROR IF NO CARRY
2854 011226 102001      BVC      3$      ;ERROR IF NO OVERFLOW
2855 011230 100401      BMI      4$      ;BRANCH IF GOOD
2856 011232 104001      3$:      ERROR   +1      ;CPU ERROR
2857                                ;CC SHOULD = 1000
2858 011234 005014      4$:      CLR      (R4)
2859 011236 005214      INC      (R4)      ;400=1
2860 011240 031114      BIT      (R1),(R4) ;125252+1=0
2861 011242 001401      BEQ      6$      ;BRANCH IF GOOD
2862 011244 104001      5$:      ERROR   +1      ;CPU ERROR
2863                                ;CC SHOULD= 0100
2864 011246      6$:
2865
2866      ;
2869 011246      MM22:
2870
2871      ;      TEST MOV MODE 2,2
2872 011246 012704 000400      MOV      #400,R4      ;R4=400
2873 011252 012701 000402      MOV      #402,R1      ;R1=402
2874 011256 012714 000005      MOV      #5,(R4)      ;400=5
2875 011262 005021      CLR      (R1)+      ;402=0
2876 011264 005011      CLR      (R1)
2877 011266 005111      COM      (R1)      ;404=-1
2878 011270 005741      TST      -(R1)      ;R1=402
2879 011272 000277      SCC
2880 011274 012124      MOV      (R1)+,(R4)+ ;CC=1111
2881 011276 100403      BMI      1$      ;400=0 R4=402 R1=404
2882 011300 103002      BCC      1$      ;ERROR IF MINUS
2883 011302 102401      BVS      1$      ;ERROR IF NO CARRY
2884 011304 001401      BEQ      2$      ;ERROR IF OVERFLOW
2885 011306 104001      1$:      ERROR   +1      ;BRANCH IF GOOD
2886                                ;CPU ERROR
2887 011310 005244      2$:      INC      -(R4)      ;CC SHOULD= 0101
2888                                ;400=1 R4=400
2889 011312 061411      ADD      (R4),(R1)   ;1+ -1 =0
2890 011314 001401      BEQ      4$      ;BRANCH IF GOOD
2891 011316 104001      3$:      ERROR   +1      ;CPU ERROR
2892                                ;CC SHOULD = 0100
2893 011320      4$:
2894
2895      ;
2898 011320      MS22:
2899
2900      ;      TEST SUB MODE 2,2
2901 011320 012704 000400      MOV      #400,R4      ;R4=400
2902 011324 012701 000402      MOV      #402,R1      ;R1=402
2903 011330 012714 177760      MOV      #177760,(R4) ;400=177760
2904 011334 012711 177750      MOV      #177750,(R1) ;402=177750
2905 011340 162421      SUB      (R4)+,(R1)+ ;R1=177770
2906 011342 001403      BEQ      1$      ;R1=177770
2907 011344 102402      BVS      1$      ;ERROR IF ZERO
2908 011346 103001      BCC      1$      ;ERROR IF OVERFLOW
2909 011350 100401      BMI      1$      ;ERROR IF NO CARRY
2910 011352 104001      1$:      ERROR   +1      ;BRANCH IF GOOD
2911                                ;CPU ERROR
2912 011354 005241      2$:      INC      -(R1)      ;CC SHOULD=1000
2913 011356 162721 177771      SUB      #177771,(R1)+ ;R1=177771
2914                                ;R1=0

```


***** DOUBLE OPERAND TESTS *****

```

2914 011362 100401          BMI      3$          ;ERROR IF MINUS
2915 011364 001401          BEQ      4$          ;BRANCH IF GOOD
2916 011366 104001          3$:      ERROR      +1          ;CPU ERROR
2917                                ;CCSHOULD = 0100
2918 011370          4$:
2919
2920          ;
2936 011370          M8B22:
2937
2938          ;      TEST BIC, BICB, BIS, BISB MODE 2,2
2939 011370 012704 000400      MOV      #400,R4          ;R4=400
2940 011374 012701 000402      MOV      #402,R1          ;R1=402
2941 011400 012702 000404      MOV      #404,R2          ;R2=404
2942 011404 012714 141401      MOV      #141401,(R4)     ;400=303 001
2943 011410 012711 177405      MCV      #177405,(R1)     ;402=377 005
2944 011414 012722 000070      MOV      #70,(R2)+        ;404=2070
2945 011420 012722 177777      MOV      #-1,(R2)+        ;406=-1
2946 011424 042421          BIC      (R4)+,(R1)+      ;402=074004
2947 011426 001401          BEQ      1$          ;ERROR IF ZERO
2948 011430 100001          BPL      2$          ;BRANCH IF GOOD
2949                                ;CC SHOULD = 1000
2950 011432 104001          1$:      ERROR      +1          ;CPU ERROR
2951 011434 052421          2$:      BIS      (R4)+,(R1)+    ;404=074074
2952 011436 142421          BICB     (R4)+,(R1)+    ;406=074
2953 011440 005301          DCC      R1              ;R4=405 R1=406
2954 011442 152421          BISB     (R4)+,(R1)+    ;406=-1 R4=406 R1=407
2955 011444 100401          BMI      4$          ;BRANCH IF GOOD
2956                                ;406 SHOULD=-1
2957 011446 104001          3$:      ERROR      +1          ;CPU ERROR
2958 011450 005214          4$:      INC      (R4)         ;406 SHOULD=0
2959 011452 001401          BEQ      6$          ;BRANCH IF GOOD
2960                                ;ERROR! 406 NE 0
2961 011454 104001          5$:      ERROR      +1          ;CPU ERROR
2962 011456          6$:
2963
2964          ;
2967 011456          M8C22:
2968
2969          ;      TEST BIT, CMP MODE 2,2
2970 011456 012704 000400      MOV      #400,R4          ;R4=400
2971 011462 012701 000402      MOV      #402,R1          ;R1=402
2972 011466 012714 125252      MOV      #125252,(R4)     ;400=125252
2973 011472 012721 100001      MOV      #100001,(R1)+    ;402=100001
2974 011476 012711 100002      MOV      #100002,(R1)     ;404=100002
2975 011502 005741          TST      -(R1)           ;R1=402
2976 011504 132421          BITB     (R4)+,(R1)+    ;**ANDED RESULT= 000
2977 011506 100401          BMT      1$          ;ERROR IF MINUS
2978 011510 001401          BEQ      2$          ;BRANCH IF GOOD
2979 011512 104001          1$:      ERROR      +1          ;CPU ERROR
2980                                ;CC SHOULD = 0100
2981 011514 132124          2$:      BITB     (R1)+,(R4)+    ;** ANDED RESULT = 200
2982 011516 001401          BEQ      3$          ;ERROR IF EQUAL
2983 011520 100401          BMI      4$          ;BRANCH IF GOOD
2984 011522 104001          3$:      ERROR      +1          ;CPU ERROR
2985                                ;CC SHOULD= 1000 R4=402 R1=404
2986 011524 022421          4$:      CMP      (R4)+,(R1)+    ;RESULT =+1
2987 011526 001402          BEQ      5$          ;ERROR IF EQUAL
    
```


***** DOUBLE OPERAND TESTS *****

```

3049 011700 104001      3$:  ERROR  +1          ;CPU ERROR
3050 011702 005724      4$:  TST    (R4)+          ;R4=402
3051 011704 005201          INC    R1              ;R1=403
3052 011706 124441      CMPB   -(R4),-(R1)     ;173 - 173=0
3053 011710 001401      BEQ    6$              ;BRANCH IF GOOD
3054                                ;BAD COMPARE
3055 011712 104001      5$:  ERROR  +1          ;CPU ERROR
3056 011714      6$:
3057                                ;
3060 011714      MA55:
3061                                ;
3062                                ; TEST ADD MODE 5,5
3063 011714 012704 000400  MOV    #400,R4          ;400=1
3064 011720 012724 000001  MOV    #1,(R4)+         ;402=-2
3065 011724 012724 177776  MOV    '-2,(R4)+       ;404=400
3066 011730 012724 000400  MOV    #400,(R4)+      ;406=402 R4=406
3067 011734 012714 000402  MOV    #402,(R4)       ;R1=410
3068 011740 012701 000410  MOV    #410,R1
3069 011744 065451      ADD    8-(R4),8-(R1)   ;1+ -2= -1
3070 011746 001402      BEQ    1$              ;ERROR IF ZERO
3071 011750 100001      BPL    1$              ;ERROR IF PLUS
3072 011752 103001      BCC    2$              ;BRANCH IF GOOD
3073 011754 104001      1$:  ERROR  +1          ;CPU ERROR
3074                                ;BAD ADD
3075 011756 062704 000004  2$:  ADD    #4,R4          ;R4=410
3076 011762 065154      ADD    8-(R1),8-(R4)   ;-1 + 1 = 0
3077 011764 001401      BEQ    4$              ;BRANCH IF GOOD
3078 011766 104001      3$:  ERROR  +1          ;CPU ERROR
3079                                ;CC SHOULD= 0100 R4=406 R1=402
3080 011770      4$:
3081                                ;
3082                                ;
3085                                ;
3086                                ;
3087                                ;-----
3088                                ;TEST DOP BIT(8) MODE 6.6
3089                                ;
3090                                ;
3091                                ;
3092                                ;
3093 011770      MB66:
3094                                ;
3095                                ; TEST BIT, BITB MODE 6.6
3096 011770 005004      CLR    R4              ;R4=0
3097 011772 012701 000400  MOV    #400,R1          ;
3098 011776 012721 125252  MOV    #125252,(R1)+   ;400=125252
3099 012002 012721 000001  MOV    #1,(R1)+        ;402=1
3100 012006 012721 100000  MOV    #100000,(R1)+   ;404=100000 R1=406
3101 012012 036461 000400 177774  BIT    400(R4),-4(R1)  ;(400)+(402)=0
3102 012020 001401      BEQ    2$              ;BRANCH IF GOOD
3103                                ;CC SHOULD = 0100
3104 012022 104001      1$:  ERROR  +1          ;CPU ERROR
3105 012024 136461 000405 177772  2$:  BITB   405(R4), 6(R1) ;(405)+(400)=200
3106 012032 001401      BEQ    3$              ;ERROR IF ZERO
3107 012034 100401      BMI    4$              ;BRANCH IF GOOD
3108                                ;CC SHOULD = 1000
3109 012036 104001      3$:  ERROR  +1          ;CPU ERROR

```

***** DOUBLE OPERAND TESTS *****

```

3110 012040      4:
3111
3112            ;
3115 012040      MS77:
3116
3117            ; TEST SUB MODE 7,7
3118 012040 012704 000400      MOV      #400,R4      ;
3119 012044 005001      CLR      R1
3120 012046 012724 177776      MOV      #-2,(R4)+      ;400= 2
3121 012052 012724 177777      MOV      #-1,(R4)+      ;402=-1
3122 012056 012724 000400      MOV      #400,(R4)+      ;404=400 R4=406
3123 012062 012711 000402      MOV      #402,(R1)      ;0=402
3124 012066 005201      TNC      R1      ;R1=1
3125 012070 167471 177372 000403      SUB      @-406(R4),@403(R1)      ;-2 - -1 = -1
3126 012076 001401      BEQ      1#      ;ERROR IF ZERO
3127 012100 100401      BMI      2#      ;BRANCH IF GOOD
3128
3129 012102 104001      1#:      ERROR      +1      ;CPU ERROR
3130 012104 167174 177777 177776 2#:      SUB      @-1(R1),@-2(R4)      ; 1 - -1 = 0
3131 012112 001401      BEQ      4#      ;BRANCH IF GOOD
3132 012114 104001      3#:      ERROR      +1      ;CPU ERROR
3133
3134 012116 005067 165656      4#:      CLR      0      ;ERROR ON SUBTRACT 400=0
3135 012122 005067 165654      CLR      2      ;RESTORE VECTORS
3136
3137
3138            ;
3141 012126      MRL0:
3142
3143            ; TEST ROL, ROLB MODE 0
3144 012126 012704 125252      MOV      #125252,R4      ;R4=125252
3145 012132 000277      SCC      ;CC=1111
3146 012134 006104      ROL      R4      ;R4=052525 WITH CARRY SET
3147 012136 102004      BVC      1#      ;ERROR IF V CLEAR
3148 012140 103003      BCC      1#      ;ERROR IF CARRY CLEAR
3149 012142 022704 052525      CMP      #052525,R4      ;SEE IF R0 = EXPECTED
3150 012146 001401      BEQ      2#      ;ERROR IF R4 NE EXPECTD
3151 012150 104001      1#:      ERROR      +1      ;CPU ERROR
3152
3153 012152 012704 125252      2#:      MOV      #125252,R4      ;ROL FAILED, CC SHOULD=0011
3154 012156 000257      CCC      ;R4=125252
3155 012160 106104      ROLB     R4      ;CC=0000
3156 012162 103005      BCC      3#      ;ROTATE EVEN BYTE
3157 012164 102004      BVC      3#      ;ERROR IF NO CARRY
3158 012166 100403      BMI      3#      ;ERROR IF NO OVERFLOW
3159 012170 022704 125124      CMP      #125124,R4      ;ERROR IF MINUS
3160 012174 001401      BEQ      4#      ;SEE IF R4 = EXPECTED
3161 012176 104001      3#:      ERROR      +1      ;BRANCH IF GOOD
3162
3163 012200      4#:      ;CPU ERROR
3164
3165            ;
3168 012200      MRLB1:
3169
3170            ; TEST ROL, POLB MODE 1
3171 012200 005004      CLR      R4      ;R4=0
3172 012202 012714 052525      MOV      #52525,(R4)      ;0=52525

```

***** DOUBLE OPERAND TESTS *****

```

3173 012206 006114      ROL      (R4)          ;**TEST INSTRUCTION, 0=125252
3174 012210 100005      BPL      1$          ;ERROR IF PLUS
3175 012212 102004      BVC      1$          ;ERROR IF NO OVERFLOW
3176 012214 103403      BCS      1$          ;ERROR IF CARRY
3177 012216 021427 125252  CMP      (R4),#125252 ;SEE IF R4=EXPECTED
3178 012222 001401      BEQ      2$          ;BRANCH IF GOOD
3179                                     ;BAD ROL ,CC SHOULD=1010
3180 012224 104001      1$:      ERROR      +1      ;CPU ERROR
3181 012226 012714 125252  2$:      MOV      #125252,(R4) ;0=125252
3182 012232 005204      INC      R4          ;R4=1
3183 012234 000277      SCC      ;CC=1111
3184 012236 106114      ROLB     (R4)          ;**TEST INSTRUCTION
3185 012240 100406      BMI      3$          ;ERROR IF RESULT IS POSITIVE
3186 012242 103005      BCC      3$          ;ERROR IF NO CARRY
3187 012244 102004      BVC      3$          ;ERROR IF V CLEAR
3188 012246 005304      DEC      R4          ;R4=0
3189 012250 022714 052652  CMP      #52652,(R4) ;ERROR IF 0 NE EXPECTED
3190 012254 001401      BEQ      4$          ;BRANCH IF GOOD
3191 012256 104001      3$:      ERROR      +1      ;CPU ERROR
3192                                     ;EAD ROLB ODD BYTE,CC SHOULD=1011
3193 012260      4$:
3194
3195      ;
3198 012260      MRL2:
3199
3200      ;      TEST ROL, ROLB MODE 2
3201 012260 005004      CLR      R4          ;R4=0
3202 012262 012714 100000  MOV      #100000,(R4) ;0=100000
3203 012266 000257      CCC      ;CC=0000
3204 012270 006124      ROL      (R4)+       ;**TEST INSTRUCTION
3205 012272 103002      BCC      1$          ;ERROR IF NO CARRY
3206 012274 102001      BVC      1$          ;ERROR IF NO OVERFLOW
3207 012276 001401      BEQ      2$          ;BRANCH IF GOOD
3208                                     ;ROL FAILED ,CCSHOULD= 0100
3209 012300 104001      1$:      ERROR      +1      ;CPU ERROR
3210 012302 005304      2$:      DEC      R4
3211 012304 005304      DEC      R4
3212 012306 001012      BNE      3$          ;ERROR IN AUTO-DEC
3213 012310 012714 004040  MOV      #4040,(R4) ;0=4040
3214 012314 000241      CLC
3215 012316 106124      ROLB     (R4)+       ;**TEST INSTURCTION
3216 012320 103405      BCS      3$          ;ERROR IF CARRY SET
3217 012322 102404      BVS      3$          ;ERROR IF V
3218 012324 005304      DEC      R4
3219 012326 022714 004100  CMP      #04100,(R4) ;SEE IF 0= EXPECTED RESULT
3220 012332 001401      BEQ      4$          ;BRANCH IF GOOD
3221 012334 104001      3$:      ERROR      +1      ;CPU ERROR
3222                                     ;BAD ROL
3223 012336      4$:
3224
3225      ;
3228 012336      MRL3:
3229
3230      ;      TEST ROL, ROLB MODE 3
3231 012336 005004      CLR      R4          ;R4=0
3232 012340 012714 052525  MOV      #052525,(R4) ;0=52525
3233 012344 000277      SCC      ;CC=1111
    
```

***** DOUBLE OPERAND TESTS *****

```

3234 012346 006137 000000      ROL      000      ;**TEST INSTRUCTION MODE 3 WITH PC
3235 012352 100005              BPL      1$      ;ERROR IF PLUS
3236 012354 102004              BVC      1$      ;ERROR IF NO OVERFLOW
3237 012356 103403              BCS      1$      ;ERROR IF CARRY
3238 012360 022714 125253      CMP      0125253,(R4) ;COMPARE RESULT WITH EXPECTED
3239 012364 001401              BEQ      2$      ;BRANCH IF GOOD
3240
3241 012366 104001              1$:      EROR      +1      ;CPU ERROR
3242 012370 012714 125252      2$:      MOV      0125252,(R4) ;O=125252
3243 012374 000261              SEC
3244 012376 106137 000000      ROLB     000      ;CC= --1
3245 012402 100402              BMI      3$      ;**TEST INSTRUCTION
3246 012404 103001              BCC      3$      ;ERROR IF MINUS
3247 012406 102401              BVS      4$      ;ERROR IF NO CARRY
3248 012410 104001              3$:      ERROR      +1      ;BRANCH IF OVERFLOW
3249
3250 012412              4$:
3251
3252
3255 012412              ;MRL4:
3256
3257              ;
3258 012412 005001              TEST ROL MODE 4
3259 012414 012704 000002      CLR      R1      ;R1=0
3260 012420 012711 054321      MOV      02,R4   ;R4=2
3261 012424 000277              MOV      054321,(R1) ;O=54321
3262 012426 006144              SCC
3263 012430 100007              ROL      -(R4)   ;CC=1111
3264 012432 102006              BPL      1$      ;**TEST INSTRUCTION
3265 012434 103405              BVC      1$      ;ERROR IF PLUS
3266 012436 022711 130643      BCS      1$      ;ERROR IF NO OVERFLOW
3267 012442 001002              BCC      1$      ;ERROR IF CARRY
3268 012444 005704              CMP      0130643,(R1) ;SEE IF EXPECTED RESULT
3269 012446 001401              BNE      1$      ;BRANCH IF ROL FAILED
3270
3271 012450 104001              TST      R4      ;SEE IF AUTO-DEC WORKED
3272 012452              1$:      EROR      +1      ;BRANCH IF GOOD
3273
3274
3277 012452              2$:
3278
3279              ;MRL5:
3280 012452 005004              ;
3281 012454 012714 000400      TEST ROL MODE 5
3282 012460 012734 123456      CLR      R4      ;R4=0
3283 012464 000277              MOV      0400,(R4) ;O=400
3284 012466 006154              MOV      0123456,0(R4) ;400=123465, R4=2
3285 012470 100410              SCC
3286 012472 103007              ROL      0-(R4)  ;CC=1111
3287 012474 102006              BMI      1$      ;**TEST INSTRUCTION
3288 012476 005704              BCC      1$      ;ERROR IF RESULT IS MINUS
3289 012500 001004              BVC      1$      ;ERROR IF NO CARRY
3290 012502 022737 047135 000400      BCS      1$      ;ERROR IF NO OVERFLOW
3291 012510 001401              TST      R4      ;SEE IF AUTO-DEC WORKED
3292
3293 012512 104001              BNE      1$      ;ERROR OF AUTO-DEC
3294 012514              1$:      EROR      +1      ;SEE IF CORRECT RESULT
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```


***** DOUBLE OPERAND TESTS *****

```

3295
3296
3299 012514      MRL6:
3300
3301      ;      TEST ROL MODE 6
3302 012514 012704 000400      MOV      #400,R4      ;R4=400
3303 012520 005001      CLR      R1      ;R1=0
3304 012522 012711 032525      MOV      #32525,(R1) ;O=32525
3305 012526 000277      SCC
3306 012530 006164 177400      ROL      -400(R4)      ;**TEST INSTRUCTION
3307 012534 100405      BMI      1$      ;ERROR IF MINUS
3308 012536 103404      BCS      1$      ;ERROR IF CARRY
3309 012540 102403      BVS      1$      ;ERROR IF OVERFLOW
3310 012542 022711 065253      CMP      #65253,(R1) ;SEE IF CORRECT RESULT
3311 012546 001401      BEQ      2$      ;BRANCH IF GOOD
3312
3313 012550 104001      1$:      ERROR      +1      ;CPU ERROR
3314 012552      2$:
3315
3316      ;
3319 012552      MRL7:
3320
3321      ;      TEST ROL MODE 7
3322 012552 012704 000400      MOV      #400,R4      ;R4=400
3323 012556 005037 000402      CLR      #402      ;402=0
3324 012562 012737 100000 000000      MOV      #100000,#0 ;O=100000
3325 012570 006174 000002      ROL      #2(R4)      ;**TEST INSTRUCTION
3326 012574 100406      BMI      1$      ;ERROR IF MINUS
3327 012576 001005      BNE      1$      ;ERROR IF NOT ZERO
3328 012600 103004      BCC      1$      ;ERROR IF NO CARRY
3329 012602 102003      BVC      1$      ;ERROR IF NO OVERFLOW
3330 012604 005737 000000      TST      #0      ;CHECK RESULT
3331 012610 001401      BEQ      2$      ;BRANCH IF GOOD
3332
3333 012612 104001      1$:      ERROR      +1      ;CPU ERROR
3334 012614      2$:
3335
3336      ;
3397 012614      MSW37:
3398
3399      ;      TEST SWAB MODE 37
3400 012614 012704 000400      MOV      #400,R4      ;R4=400
3401 012620 012714 040700      MOV      #40700,(R4) ;400= 101 300
3402 012624 000337 000400      SWAB     #400      ;400 SHOULD = 300 101
3403 012630 100406      BMI      1$      ;ERROR IF MINUS
3404 012632 022714 140101      CMP      #140101,(R4) ;SEE IF EXPECTED RESULT
3405 012636 001003      BNE      1$      ;BRANCH IF BAD
3406 012640 000337 000400      SWAB     #400      ;400=101 300
3407 012644 100401      BMI      2$      ;BRANCH IF GOOD
3408
3409 012646 104001      1$:      ERROR      +1      ;ERROR! BAD SWAB MODE 37
3410 012650      2$:
3411
3412      ;
3415 012650      MRR0:
3416
3417      ;      TEST ROR MODE 0

```

***** DOUBLE OPERAND TESTS *****

```

3418 012650 012704 052525      MOV    #52525,R4          ;R4=52525
3419 012654 000257            CCC                    ;CC=0000
3420 012656 006004            ROR    R4                ;R4 SHOULD = 25252 WITH CARRY
3421 012660 103003            BCC    1$                ;ERROR IF NO CARRY
3422 012662 022704 025252      CMP    #25252,R4        ;SEE IF R4= EXPECTED
3423 012666 001401            BEQ    2$                ;BRANCH IF GOOD
3424                                ;ROR MODE 0 FAILED
3425 012670 104001            1$:   ERROR    +1        ;CPU ERROR
3426 012672                    2$:
3427
3428                                ;
3431 012672                    MRRB1:
3432
3433                                ;
3434 012672 005004            ;   TEST RORB MODE 1
3435 012674 012714 000001      CLR    R4                ;R4=0
3436 012700 000277            MOV    #1,(R4)          ;0=1
3437 012702 106014            SCC                    ;CC=1111
3438 012704 103004            RORB   (R4)             ;0=000200, NO C
3439 012706 100003            BCC    1$                ;ERROR IF NO CARRY
3440 012710 022714 000200      BPL    1$                ;ERROR IF PLUS
3441 012714 001401            CMP    #200,(R4)        ;CHECK RESULT
3442 012716 104001            BEQ    2$                ;BRANCH IF GOOD
3443                                1$:   ERROR    +1        ;CPU ERROR
3444                                2$:   ;BAD RESULT
3445
3446                                ;
3448
3450 012720                    MJ:
3451
3452                                ;
3453 012720 012737 000001 003072 ;   TEST JMP - ALL MODES
3454 012726 012701 012772      MOV    #1,@#SEQ        ;SETUP TEST SEQUENCER
3455 012732 000111            MOV    #MJU1,R1        ;SET MODE 1 JUMP ADDRESS
3456 012734 023727 003072 000002 MJU2:  JMP    (R1)             ;*JMP MODE 1
3457 012742 001401            CMP    @#SEQ,#2        ;CHECK FOR CORRECT SEQUENCE
3458                                BEQ    MJU2A            ;BRANCH IF GOOD
3459 012744 104001            1$:   ERROR    +1        ;MODE 2 JUMP FAILED
3460 012746 020127 012736      MJU2A:  CMP    R1,#MJU2+2   ;CPU ERROR
3461 012752 001401            BEQ    MJU2B            ;CHECK FOR AUTO-INCREMENT
3462 012754 104001            2$:   ERROR    +1        ;BRANCH IF GOOD
3463                                ;CPU ERROR
3464 012756 005237 003072      MJU2B:  INC    @#SEQ        ;AUTO-INCR FAILED
3465 012762 012701 012770      MOV    #MJ2,R1         ;UPDATE TEST SEQUENCER
3466 012766 000131            JMP    @(R1)+           ;SETUP MODE 3 JUMP
3467 012770 013016            MJ2:   .WORD    MJU3    ;JUMP MODE 3
3468 012772 023727 003072 000001 MJU1:   CMP    @#SEQ,#1    ;MODED 3 DETINATION
3469 013000 001401            BEQ    MJU1A            ; TEST FOR CORRECT SEQUENCE
3470 013002 104001            3$:   ERROR    +1        ;BRANCH IF GOOD
3471                                ;CPU ERROR
3472 013004 005237 003072      MJU1A:  INC    @#SEQ        ;JMP OUT OF SEQUENCE
3473 013010 012701 012734      MOV    #MJU2,R1        ;UPDATE SEQUENCE
3474 013014 000121            JMP    (R1)+           ;SETUP MODE 2 DESTINATION
3475 013016 023727 003072 000003 MJU3:   CMP    @#SEQ,#3    ;JUMP MODE 2
3476 013024 001401            BEQ    MJU3A            ; TEST FOR CORRECT SEQUENCE
3477 013026 104001            4$:   ERROR    +1        ;BRANCH IF GOOD
3478                                ;CPU ERROR
                                ;JMP OUT OF SEQUENCE

```

Z

***** DOUBLE OPERAND TESTS *****

```

3479 013030 022701 012772 MJU3A: CMP #MJ2+2,R1 ; TEST AUTO-INCREMENT
3480 013034 001401 BEQ MJU3B ;BRANCH IF GOOD
3481 013036 104001 5$: ERROR +1 ;CPU ERROR
3482 ;AUTO-INCREMENT FAILED MODE 3
3483 013040 005237 003072 MJU3B: INC @#SEQ ;UPDATE SEQUENCER
3484 013044 012701 013114 MOV #MJU4+2,R1 ;SETUP DESTINATION MODE 4
3485 013050 000141 JMP -(R1) ;EXECUTE JUMP MODE 4
3486 013052 000000 MJU5: HALT
3487 013054 022701 013150 CMP #MJ5,R1 ;CHECK AUTO-DECREMENT
3488 013060 001401 BEQ MJU5A ;BRANCH IF GOOD AUTO-DEC
3489 013062 104001 6$: ERROR +1 ;CPU ERROR
3490 ;AUTO-DEC FAILED MODE 5
3491 013064 023727 003072 000005 MJU5A: CMP @#SEQ,#5 ; TEST CORRECT SEQUENCE
3492 013072 001401 BEQ MJU5B ;BRANCH IF GOOD SEQUENCE
3493 013074 104001 7$: ERROR +1 ;CPU ERROR
3494 ;JMP OUT OF SEQUENCE
3495 013076 005237 003072 MJU5B: INC @#SEQ ;UPDATE SEQUENCE COUNT
3496 013102 012701 013145 MOV #MJU6-5,R1 ;SETUP DESTINATION MODE6
3497 013106 000161 000005 JMP +5(R1) ;JUMP MODE 6
3498 ;;
3499 013112 000240 MJU4: NOP
3500 013114 022701 013112 CMP #MJU4,R1 ; TEST AUTO DECR
3501 013120 001401 BEQ MJU4A ;BRANCH IF GOOD
3502 013122 104001 8$: ERROR +1 ;CPU ERROR
3503 ;MODE 4 AUTO-DEC FAILED
3504 013124 023727 003072 000004 MJU4A: CMP @#SEQ,#4 ; TEST FOR CORRECT SEQUENCE
3505 013132 001401 BEQ MJU4B ;BRANCH IF CORRECT SEQUENCE
3506 013134 104001 9$: ERROR +1 ;CPU ERROR
3507 ;INCORRECT JMP SEQUENCE
3508 013136 005237 003072 MJU4B: INC @#SEQ ;UPDATE SEQUENCE
3509 013142 012701 013152 MOV #MJ5+2,R1 ;SETUP MODE 5 POINTER
3510 013146 000151 JMP @-(R1) ;EXECUTE MODE 5 JMP
3511 ;
3512 013150 013054 MJ5: .WORD MJU5+2 ;POINTER MODE 5
3513 ;
3514 013152 022737 000006 003072 MJU6: CMP #6,@#SEQ ;CHECK FOR CORRECT SEQUENCE
3515 013160 001401 BEQ MJU6A ;BRANCH IF GOOD
3516 013162 104001 10$: ERROR +1 ;CPU ERROR
3517 ;INCORRECT SEQUENCE
3518 013164 005237 003072 MJU6A: INC @#SEQ ;UPDATE SEQUENCER
3519 013170 012701 013210 MOV #MJ7+10,R1 ;SETUP INDEX
3520 013174 000171 177770 JMP @-10(R1) ;EXECUTE MODE 7 JUMP
3521 013200 013204 MJ7: .WORD MJU7 ;POINTER FOR MODE 7
3522 013202 000000 HALT
3523 013204 022737 000007 003072 MJU7: CMP #7,@#SEQ ; TEST FOR CORRECT SEQUENCE
3524 013212 001401 BEQ MJU7E ;BRANCH IF GOOD SEQUENCE
3525 013214 104001 11$: ERROR +1 ;CPU ERROR
3526 ; TESTING OUT OF SEQUENCE
3527 013216 MJU7E:
3528 ;
3539 ;
3540 ; TEST THAT PRE-FETCH BUFFER CAN BE OVER WRITTEN
3541 013216 012701 013526 MOV #128$,R1 ;SET UP R1 WITH ADDRESS OF ERROR
3542 ;ROUTINE
3543 .REPT 32.
3545 013222 012717 MOV (PC)+,(PC) ;WRITE THE NOP OVER THE JMP INSTRUCTION
013224 000240 .WORD NOP ;NOP INSTRUCTION

```

***** DOUBLE OPERAND TESTS *****

013226	000111	30008\$:	.WORD	111	;JMP (R1)
013230	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013232	000240		.WORD	NOP	;NOP INSTRUCTION
013234	000111	30009\$:	.WORD	111	;JMP (R1)
013236	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013240	000240		.WORD	NOP	;NOP INSTRUCTION
013242	000111	30010\$:	.WORD	111	;JMP (R1)
013244	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013246	000240		.WORD	NOP	;NOP INSTRUCTION
013250	000111	30011\$:	.WORD	111	;JMP (R1)
013252	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013254	000240		.WORD	NOP	;NOP INSTRUCTION
013256	000111	30012\$:	.WORD	111	;JMP (R1)
013260	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013262	000240		.WORD	NOP	;NOP INSTRUCTION
013264	000111	30013\$:	.WORD	111	;JMP (R1)
013266	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013270	000240		.WORD	NOP	;NOP INSTRUCTION
013272	000111	30014\$:	.WORD	111	;JMP (R1)
013274	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013276	000240		.WORD	NOP	;NOP INSTRUCTION
013300	000111	30015\$:	.WORD	111	;JMP (R1)
013302	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013304	000240		.WORD	NOP	;NOP INSTRUCTION
013306	000111	30016\$:	.WORD	111	;JMP (R1)
013310	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013312	000240		.WORD	NOP	;NOP INSTRUCTION
013314	000111	30017\$:	.WORD	111	;JMP (R1)
013316	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013320	000240		.WORD	NOP	;NOP INSTRUCTION
013322	000111	30018\$:	.WORD	111	;JMP (R1)
013324	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013326	000240		.WORD	NOP	;NOP INSTRUCTION
013330	000111	30019\$:	.WORD	111	;JMP (R1)
013332	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013334	000240		.WORD	NOP	;NOP INSTRUCTION
013336	000111	30020\$:	.WORD	111	;JMP (R1)
013340	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013342	000240		.WORD	NOP	;NOP INSTRUCTION
013344	000111	30021\$:	.WORD	111	;JMP (R1)
013346	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013350	000240		.WORD	NOP	;NOP INSTRUCTION
013352	000111	30022\$:	.WORD	111	;JMP (R1)
013354	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013356	000240		.WORD	NOP	;NOP INSTRUCTION
013360	000111	30023\$:	.WORD	111	;JMP (R1)
013362	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013364	000240		.WORD	NOP	;NOP INSTRUCTION
013366	000111	30024\$:	.WORD	111	;JMP (R1)
013370	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013372	000240		.WORD	NOP	;NOP INSTRUCTION
013374	000111	30025\$:	.WORD	111	;JMP (R1)
013376	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013400	000240		.WORD	NOP	;NOP INSTRUCTION
013402	000111	30026\$:	.WORD	111	;JMP (R1)
013404	012717		MOV	(PC)+,(PC)	;WRITE THE NOP OVER THE JMP INSTRUCTION
013406	000240		.WORD	NOP	;NOP INSTRUCTION

***** DOUBLE OPERAND TESTS *****

```

013410 000111          30027$: .WORD 111          ;JMP (R1)
013412 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013414 000240          .WORD NOP          ;NOP INSTRUCTION
013416 000111          30028$: .WORD 111          ;JMP (R1)
013420 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013422 000240          .WORD NOP          ;NOP INSTRUCTION
013424 000111          30029$: .WORD 111          ;JMP (R1)
013426 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013430 000240          .WORD NOP          ;NOP INSTRUCTION
013432 000111          30030$: .WORD 111          ;JMP (R1)
013434 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013436 000240          .WORD NOP          ;NOP INSTRUCTION
013440 000111          30031$: .WORD 111          ;JMP (R1)
013442 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013444 000240          .WORD NOP          ;NOP INSTRUCTION
013446 000111          30032$: .WORD 111          ;JMP (R1)
013450 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013452 000240          .WORD NOP          ;NOP INSTRUCTION
013454 000111          30033$: .WORD 111          ;JMP (R1)
013456 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013460 000240          .WORD NOP          ;NOP INSTRUCTION
013462 000111          30034$: .WORD 111          ;JMP (R1)
013464 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013466 000240          .WORD NOP          ;NOP INSTRUCTION
013470 000111          30035$: .WORD 111          ;JMP (R1)
013472 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013474 000240          .WORD NOP          ;NOP INSTRUCTION
013476 000111          30036$: .WORD 111          ;JMP (R1)
013500 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013502 000240          .WORD NOP          ;NOP INSTRUCTION
013504 000111          30037$: .WORD 111          ;JMP (R1)
013506 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013510 000240          .WORD NOP          ;NOP INSTRUCTION
013512 000111          30038$: .WORD 111          ;JMP (R1)
013514 012717          MOV (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013516 000240          .WORD NOP          ;NOP INSTRUCTION
013520 000111          30039$: .WORD 111          ;JMP (R1)
3546 013522 000137 013530 JMP @#129$          ;JUMP OVER ERROR CALL
3547 013526          128$:          ;ERROR! PRE-FETCH BUFFER WAS NOT
3548          ;OVER WRITTEN
3549 013526 104001          ERROR +1          ;CPU ERROR
3550          ;
3551          ; NOW RESTORE THE OVER WRITTEN JMP INSTRUCTIONS FOR THE NEXT PASS.
3552          ;
3553 013530 012702 000040 129$: MOV #32,R2          ;SET UP R2 AS COUNTER
3554 013534 012703 013216 MOV #MJU7E,R3        ;SET UP R3 AS POINTER. $$$
3555 013540 062703 000010 ADD #10,R3          ;ADD OFFSET TO POINT TO 1st. JMP IN TABLE. $$$
3556 013544 012713 000111 130$: MOV #111,(R3)        ;RESTORE OVER WRITTEN JUMPS
3557 013550 062703 000006 ADD #6,R3           ;POINT TO NEXT OVER WRITTEN ADDR.
3558 013554 077205 SOB R2,130$         ;DO UNTIL R2=0
3559
3561          ;
3563 013556          MJP:
3564
3565          ; TEST JMP MODES 17,27,37,67,77
3566 013556 012737 000000 003072 MOV #0,@#SEQ        ;SETUP TEST SEQUENCER
3567 013564 000117 JMP (R7)            ;JUMP MODE 17(SHOULD BE IN LINE)

```

***** DOUBLE OPERAND TESTS *****

```

3568
3569 013566 005737 003072      ; MJP17: TST      @#SEQ      ;CHECK SEQUENCE
3570 013572 001401              BEQ      2#              ;BRANCH IF GOOD
3571                          ;              ;ERROR! BAD JUMP
3572 013574 104001              1#:      ERROR      +1      ;CPU ERROR
3573 013576 005237 003072      2#:      INC      @#SEQ      ;UPDATE SEQUENCE NUMBER
Z 3574 013602 000127 000401      JMP      @401          ;JU'IP MODE 27
3575                          ;              ;(THE @401=BR UPDATED PC+2)
3576                          ;
3577 013606 000000              ;      HALT              ;
3578                          ;
3579 013610 023727 003072 000001 MJP27:  CMP      @#SEQ, #1      ;CHECK IF CORRECT SEQUENCE
3580 013616 001401              BEQ      MJP27A        ;BRANCH IF IN SEQUENCE
3581 013620 104001              1#:      ERROR      +1      ;CPU ERROR
3582                          ;              ; TEST OUT OF SEQUENCE
3583 013622 005237 003072      MJP27A: INC      @#SEQ      ;UPDATE SEQUENCER
3584 013626 000137 013674              JMP      @#MJP37      ;JUMP MODE 37
3585 013632 023727 003072 000003 MJP67:  CMP      @#SEQ, #3      ;CHECK FOR CORRECT SEQUENCE
3586 013640 001401              BEQ      MJP67A        ;BRANCH IF IN SEQUENCE
3587 013642 104001              2#:      ERROR      +1      ;CPU ERROR
3588                          ;              ; TEST OUT OF SEQUENCE
3589 013644 005237 003072      MJP67A: INC      @#SEQ      ;UPDATE SEQUENCER
3590 013650 000257              CCC              ;INSURE ZBIT=0
3591 013652 000177 000002      JMP      @MJP67B      ;JUMP MODE 7
3592                          ;
3593 013656 000000              ;      HALT              ;
3594                          ;
3595 013660 013662              MJP67B: .WORD      MJP77      ;
3596                          ;
3597 013662 023727 003072 000004 MJP77:  CMP      @#SEQ, #4      ; TEST FOR CORRECT SEQUENCE
3598 013670 001412              BEQ      MJP77E        ;BRANCH IF IN SEQUENCE
3599 013672 104001              3#:      ERROR      +1      ;CPU ERROR
3600                          ;              ; TEST OUT OF SEQUENCE
3601 013674 023727 003072 000002 MJP37:  CMP      @#SEQ, #2      ; TEST FOR CORRECT SEQUENCE
3602 013702 001401              BEQ      MJP37A        ;BRANCH IF IN SEQUENCE
3603 013704 104001              4#:      ERROR      +1      ;CPU ERROR
3604                          ;              ; TEST OUT OF SEQUENCE
3605 013706 005237 003072      MJP37A: INC      @#SEQ      ;UPDATE SEQUENCER
3606 013712 000167 177714      JMP      MJP67        ;JUMP MODE 6
3607                          ;
3608                          ;
3609 013716              MJP77E:
3610                          ;
3611                          ;
3614 013716              MJSR:
3615                          ;
3616                          ;      TEST JSR ALL MODES
3617 013716 010637 003074              MOV      R6, @#SPS      ;SAVE STACK POINTER LOCATION
3618 013722 010637 003076              MOV      R6, @#SPSJ     ;
3619 013726 162737 000002 003076      SUB      #2, @#SPSJ     ;SPSJ = R6 AFTER DECRIMENT
3620 013734 012737 000001 003072      MOV      #1, @#SEQ      ;SETUP SEQUENCE COUNTER
3621 013742 012701 014036              MOV      @MJSR1, R1     ;SETUP INITIAL JUMP IN MODE 1
3622 013746 005004              CLR      R4              ;
3623 013750 005104              COM      R4              ;R4=-1 TO BE SAVED ON STACK
3624 013752 004411              JSR      R4, (R1)       ;JSR MODE 1
3625                          ;
3626 013754 022737 000002 003072 MJSR2:  CMP      #2, @#SEQ      ; TEST FOR CORRECT SFQUENCE

```

***** DOUBLE OPERAND TESTS *****

3627	013762	001401			BEQ	MJSR2A		;BRANCH IF GOOD
3628	013764	104001			5\$:	ERROR	+1	;CPU ERROR
3629								;MODE 2 JUMPED TO OUT OF SEQUENCE
3630	013766	023706	003076		MJSR2A:	CMP	@SPSJ,R6	;VERIFY STACK DECRIMENT
3631	013772	001006				BNE	6\$;BRANCH IF STACK INCORRECT
3632	013774	021627	125252			CMP	(R6),#125252	;VERIFY CONTENTS OF STACK
3633	014000	001003				BNE	6\$;BRANCH IF DATA ON STACK INCORRECT
3634	014002	022704	014116			CMP	@MJSR4,R4	;SEE IF CORRECT RETURN ADDRESS
3635	014006	001401				BEQ	MJSR2B	;BRANCH IF GOOD
3636	014010	104001			6\$:	ERROR	+1	;CPU ERROR
3637								;JSR MODE 2 FAILED
3638	014012	005237	003072		MJSR2B:	INC	@SEQ	;UPDATE SEQUENCE COUNTER
3639	014016	013706	003074			MOV	@SPS,R6	;RELOAD STACK POINTER
3640	014022	012701	014034			MOV	@MJSRA,R1	;SETUP JSR MODE 3
3641	014026	005004				CLR	R4	;DIFFERENT DATA TO R4
3642	014030	004431				JSR	R4,@(R1)+	;JSR MODE 3
3643	014032	000000				HALT		
3644	014034	014202			MJSRA:	.WORD	MJSR3	;LITERAL FOR JUMP MODE 3
3645								
3646	014036	022737	000001	003072	MJSR1:	CMP	#1,@SEQ	;TEST FOR CORRECT SEQUENCE
3647	014044	001401				BEQ	MJSR1A	;BRANCH IF GOOD
3648	014046	104001			7\$:	ERROR	+1	;CPU ERROR
3649								;MODE 1 JUMPED TO OUT OF SEQUENCE
3650	014050	023706	003076		MJSR1A:	CMP	@SPSJ,R6	;VERIFY STACK DECRIMENT
3651	014054	001006				BNE	8\$;BRANCH IF STACK INCORRECT
3652	014056	021627	177777			CMP	(R6),#-1	;VERIFY CONTENT OF STACK
3653	014062	001003				BNE	8\$;BRANCH IF DATA ON STACK INCORRECT
3654	014064	022704	013754			CMP	@MJSR2,R4	;SEE IF CORRECT RETURN ADDRESS
3655	014070	001401				BEQ	MJSR1B	;BRANCH IF GOOD
3656	014072	104001			8\$:	ERROR	+1	;CPU ERROR
3657								;JSR MODE 2 FAILED
3658	014074	005237	003072		MJSR1B:	INC	@SEQ	;UPDATE SEQUENCE COUNTER
3659	014100	013706	003074			MOV	@SPS,R6	;RELOAD STACK POINTER
3660	014104	012704	125252			MOV	#125252,R4	;SETUP R4 DATA
3661	014110	012701	013754			MOV	@MJSR2,R1	;SETUP MODE 2 JUMP ADDRESS
3662	014114	004421				JSR	R4,(R1)+	;JUMP MODE 2
3663								
3664								
3665	014116	022737	000004	003072	MJSR4:	CMP	#4,@SEQ	;TEST FOR CORRECT SEQUENCE
3666	014124	001401				BEQ	MJSR4A	;BRANCH IF GOOD
3667	014126	104001			9\$:	ERROR	+1	;CPU ERROR
3668								;MODE 4 JUMPED TO OUT OF SEQUENCE
3669	014130	023706	003076		MJSR4A:	CMP	@SPSJ,R6	;VERIFY STACK DECRIMENT
3670	014134	001006				BNE	10\$;BRANCH IF STACK INCORRECT
3671	014136	021627	052525			CMP	(R6),#052525	;VERIFY CONTENTS OF STACK
3672	014142	001003				BNE	10\$;BRANCH IF DATA ON STACK INCORRECT
3673	014144	022704	014264			CMP	@MJSR6,R4	;SEE IF CORRECT RETURN ADDRESS
3674	014150	001401				BEQ	MJSR4B	;BRANCH IF GOOD
3675	014152	104001			10\$:	ERROR	+1	;CPU ERROR
3676								;JSR MODE 4 FAILED
3677	014154	005237	003072		MJSR4B:	INC	@SEQ	;UPDATE SEQUENCE COUNTER
3678	014160	013706	003074			MOV	@SPS,R6	;RELOAD STACK POINTER
3679	014164	012704	000377			MOV	#377,R4	;SETUP R4 DATA
3680	014170	012701	014202			MOV	@MJSRB+2,R1	;SETUP JSR VECTOR
3681	014174	004451				JSR	R4,@-(R1)	;JSR MODE 5
3682	014176	000000				HALT		
3683	014200	014350			MJSRB:	.WORD	MJSR5	;MODE 5 VECTOR

***** DOUBLE OPERAND TESTS *****

```

3684
3685
3686 014202 022737 000003 003072 MJSR3:  CMP    #3,#SEQ          ; TEST FOR CORRECT SEQUENCE
3687 014210 001401                BEQ    MJSR3A          ; BRANCH IF GOOD
3688 014212 104001                11$:  ERROR    +1      ; CPU ERROR
3689                                ; MODE 3 JUMPED TO OUT OF SEQUENCE
3690 014214 023706 003076 MJSR3A: CMP    #SPSJ,R6    ; VERIFY STACK DECREMENT
3691 014220 001006                BNE    12$            ; BRANCH IF STACK INCORRECT
3692 014222 021627 000000                CMP    (R6),#0        ; VERIFY CONTENTS OF STACK
3693 014226 001003                BNE    12$            ; BRANCH IF DATA ON STACK INCORRECT
3694 014230 022704 014032                CMP    #MJSRA-2,R4    ; SEE IF CORRECT RETURN ADDRESS
3695 014234 001401                BEQ    MJSR3B          ; BRANCH IF GOOD
3696 014236 104001                12$:  ERROR    +1      ; CPU ERROR
3697                                ; JSR MODE 3 FAILED
3698 014240 005237 003072 MJSR3B: INC    #SEQ          ; UPDATE SEQUENCE COUNTER
3699 014244 013706 003074                MOV    #SPS,R6        ; RELOAD STACK POINTER
3700 014250 012704 052525                MOV    #052525,R4     ; SETUP R4 DATA
3701 014254 012701 014120                MOV    #MJSR4+2,R1    ; SETUP JSR VECTOR
3702 014260 000257                CCC                                ; CLEAR CONDITION CODES
3703 014262 004441                JSR    R4,-(R1)       ; JSR MODE 4
3704
3705
3706 014264 022737 000006 003072 MJSR6:  CMP    #6,#SEQ          ; TEST FOR CORRECT SEQUENCE
3707 014272 001401                BEQ    MJSR6A          ; BRANCH IF GOOD
3708 014274 104001                13$:  ERROR    +1      ; CPU ERROR
3709                                ; MODE 6 JUMPED TO OUT OF SEQUENCE
3710 014276 023706 003076 MJSR6A: CMP    #SPSJ,R6    ; VERIFY STACK DECREMENT
3711 014302 001006                BNE    14$            ; BRANCH IF STACK INCORRECT
3712 014304 021627 123456                CMP    (R6),#123456   ; VERIFY CONTENTS OF STACK
3713 014310 001003                BNE    14$            ; BRANCH IF DATA ON STACK INCORRECT
3714 014312 022704 014432                CMP    #MJSR7,R4      ; SEE IF CORRECT RETURN ADDRESS
3715 014316 001401                BEQ    MJSR6B          ; BRANCH IF GOOD
3716 014320 104001                14$:  ERROR    +1      ; CPU ERROR
3717                                ; JSR MODE 6 FAILED
3718 014322 005237 003072 MJSR6B: INC    #SEQ          ; UPDATE SEQUENCE COUNTER
3719 014326 013706 003074                MOV    #SPS,R6        ; RELOAD STACK POINTER
3720 014332 012704 177773                MOV    #-5,R4         ; SETUP R4 DATA
3721 014336 012701 014356                MOV    #MJSRC+10,R1   ; SETUP JSR VECTOR
3722 014342 004471 177770                JSR    R4,#-10(R1)    ; JSR MODE 7
3723 014346 014432 MJSRC:  .WORD  MJSR7      ; JSR VECTOR
3724
3725
3726 014350 022737 000005 003072 MJSR5:  CMP    #5,#SEQ          ; TEST FOR CORRECT SEQUENCE
3727 014356 001401                BEQ    MJSR5A          ; BRANCH IF GOOD
3728 014360 104001                15$:  ERROR    +1      ; CPU ERROR
3729                                ; MODE 5 JUMPED TO OUT OF SEQUENCE
3730 014362 023706 003076 MJSR5A: CMP    #SPSJ,R6    ; VERIFY STACK DECREMENT
3731 014366 001006                BNE    16$            ; BRANCH IF STACK INCORRECT
3732 014370 021627 000377                CMP    (R6),#377      ; VERIFY CONTENTS OF STACK
3733 014374 001003                BNE    16$            ; BRANCH IF DATA ON STACK INCORRECT
3734 014376 022704 014176                CMP    #MJSRB-2,R4    ; SEE IF CORRECT RETURN ADDRESS
3735 014402 001401                BEQ    MJSR5B          ; BRANCH IF GOOD
3736 014404 104001                16$:  ERROR    +1      ; CPU ERROR
3737                                ; JSR MODE 5 FAILED
3738 014406 005237 003072 MJSR5B: INC    #SEQ          ; UPDATE SEQUENCE COUNTER
3739 014412 013706 003074                MOV    #SPS,R6        ; RELOAD STACK POINTER
3740 014416 012704 123456                MOV    #123456,R4     ; SETUP DATA IN R4

```

***** DOUBLE OPERAND TESTS *****

```

3741 014422 012701 014274      MOV    #MJSR6+10,R1      ;SETUP JSR VECTOR
3742 014426 004461 177770      JSR    R4,-10(R1)      ;JUMP MODE 6
3743
3744
3745 014432 022737 000007 003072 MJSR7:  CMP    #7,#0SEQ      ; TEST FOR CORRECT SEQUENCE
3746 014440 001401              BEQ    MJSR7A          ;BRANCH IF GOOD
3747 014442 104001              17$:  ERROR    +1      ;CPU ERROR
3748                          ;MODE 7 JUMPED TO OUT OF SEQUENCE
3749 014444 023706 003076      MJSR7A: CMP    #0SPSJ,R6   ;VERIFY STACK DECRIMENT
3750 014450 001006              BNE    18$            ;BRANCH IF STACK INCORRECT
3751 014452 021627 177773      CMP    (R6),#-5       ;VERIFY CONTENTS OF STAACK
3752 014456 001003              BNE    18$            ;BRANCH IF DATA ON STACK INCORRECT
3753 014460 022704 014346      CMP    #MJSR5-2,R4    ;SEE IF CORRECT RETURN ADDRESS
3754 014464 001401              BEQ    MJSR7E          ;BRANCH IF GOOD
3755 014466 104001              18$:  ERROR    +1      ;CPU ERROR
3756                          ;JSR MODE 7 FAILED
3757 014470
3758 014470 013706 003074      MJSR7E: MOV    #0SPS,R6   ;REPLACE STACK
3759
3760
3763 014474                      ;MJRA:
3764
3765
3766 014474 012737 000001 003072      ; TEST JSR MODES 27, 37, 67, 77
3767 014502 010637 003074      MOV    #1,#0SEQ      ;SETUP SEQUENCER
3768 014506 010637 003076      MOV    R6,#0SPS      ;SAVE STACK ADDRESS
3769 014512 162737 000002 003076      MOV    R6,#0SPSJ     ;SAVE STACK DECRIMENT ADDRESS
3770 014520 012704 177777      SUB    #2,#0SPSJ     ;
3771 014524 004427 000240      MOV    #-1,R4        ;SETUP R4 DATA
Z 3772                      JSR    R4,#240       ;EXECUTE A JSR MODE 27
3773 014530 022737 000001 003072 MJR27:  CMP    #1,#0SEQ      ;VERIFY COERRECT TEST SEQUENCE
3774 014536 001011              BNE    1$            ;INCORRECT TEST SEQUENCE
3775 014540 023706 003076      CMP    #0SPSJ,R6     ;VERIFY STACK POINTER
3776 014544 001006              BNE    1$            ;
3777 014546 021627 177777      CMP    (R6),#-1     ;VERIFY R4 GOT LOADED ON THE STACK
3778 014552 001003              BNE    1$            ;BRANCH IF INCORRECT STACK CONTENTS
3779 014554 020427 014530      CMP    R4,#MJR27     ;VEPIFY CORRECT RETURN ADDRESS
3780 014560 001401              BEQ    MJR27A        ;BRANCH IF GOOD RETURN ADDRESS . STACK
3781 014562 104001              1$:  ERROR    +1      ;CPU ERROR
3782                          ;MODE 27 FAILED
3783 014564 005237 003072      MJR27A: INC    #0SEQ      ;UPDATE SEQUENCER
3784 014570 012704 152525      MOV    #152525,R4   ;SETUP R4 TEST DATA
3785 014574 013706 003074      MOV    #0SPS,R6     ;RESET STACK POINTER
3786 014600 004437 014666      JSR    R4,#MJR37    ;JSR MODE 37
3787 014604 000000      MJR27B: HALT
3788
3789 014606 023727 003072 000003 MJR67:  CMP    #0SEQ,#3      ;VERIFY TEST SEQUENCE
3790 014614 001011              BNE    2$            ;INCORRECT TEST SEQUENCE
3791 014616 023706 003076      CMP    #0SPSJ,R6     ;VERIFY STACK DECRIMENT
3792 014622 001006              BNE    2$            ;INCORRECT STACK DECRIMENT
3793 014624 021627 000125      CMP    (R6),#125    ;VERFFIFY STACK WAS LOADED
3794 014630 001003              BNE    2$            ;
3795 014632 020427 014742      CMP    R4,#MJR77    ;VERIFY RETURN ADDRESS
3796 014636 001401              BEQ    MJR67A        ;BRANCH IF GOOD
3797 014640 104001              2$:  ERROR    +1      ;CPU ERROR
3798                          ;MODE 67 FAILED
3799 014642 005237 003072      MJR67A: INC    #0SEQ      ;UPDATE SEQUENCER

```


***** DOUBLE OPERAND TESTS *****

```

3859                                     ;ERROR! RTS NOT EXECUTED
3860 015046 021506          1$:  CMP      (R5),R6          ;IS R6=31506?
3861 015050 001401          BEQ      RTSE          ;IF IT IS THEN GO TO END OF TEST
3862 015052 104001          ERROR    +1          ;CPU ERROR
3863                                     ;ERROR! WRONG ADDR IN R6
3864 015054 010106          RTSE:  MOV      R1,PC          ;RESTORE STACK
3865
3868 015056          TSMMUO:
3869          ;
3870 015056 012737 040000 177776      ;
3871 015064 012706 177777      MOV      #40000,#177776      ;SET PS TO SUP MODE
3872 015070 022706 177777      MOV      #177777,R6          ;INIT SUP STACK TO ALL ONES
3873 015074 001401          CMP      #177777,R6          ;ARE ALL BITS SET
3874                                     BEQ      1$          ;YES GO ON
3875 015076 104001          ERROR    +1          ;NO GO TO ERROR
3876 015100 005006          1$:  CLR      R6          ;CPU ERROR
3877 015102 022706 000000      CMP      #0,R6          ;SET SUP STACK TO ALL ZEROES
3878 015106 001401          BEQ      2$          ;ARE ALL BITS CLEARED
3879                                     ;YES GO ON
3880 015110 104001          ERROR    +1          ;NO GO TO ERROR
3881 015112 012706 125252      2$:  MOV      #125252,R6          ;CPU ERROR
3882 015116 022706 125252      CMP      #125252,R6          ;SET SUP STACK TO ALTERNATING PATTERN
3883 015122 001401          BEQ      3$          ;IS SUP SP CORRECT
3884                                     ;YES GO ON
3885 015124 104001          ERROR    +1          ;NO GO TO ERROR
3886 015126 012706 052525      3$:  MOV      #52525,R6          ;CPU ERROR
3887 015132 022706 052525      CMP      #52525,R6          ;SET SUP STACK TO ALTERNATING PATTERN
3888 015136 001401          BEQ      4$          ;IS SUP SP CORRECT
3889                                     ;YES GO ON
3890 015140 104001          ERROR    +1          ;NO GO TO ERROR
3891 015142 012706 000700      4$:  MOV      #700,R6          ;CPU ERROR
3892 015146 012737 140000 177776      MOV      #140000,#177776      ;SETUP SUP SP
3893 015154 012706 177777      MOV      #177777,R6          ;SET PS TO USER MODE
3894 015160 022706 177777      CMP      #177777,R6          ;INIT USER STACK TO ALL ONES
3895 015164 001401          BEQ      5$          ;ARE ALL BITS SET
3896                                     ;YES GO ON
3897 015166 104001          ERROR    +1          ;NO GO TO ERROR
3898 015170 005006          5$:  CLR      R6          ;CPU ERROR
3899 015172 022706 000000      CMP      #0,R6          ;SET USER STACK TO ALL ZEROES
3900 015176 001401          BEQ      6$          ;ARE ALL BITS CLEARED
3901                                     ;YES GO ON
3902 015200 104001          ERROR    +1          ;NO GO TO ERROR
3903 015202 012706 125252      6$:  MOV      #125252,R6          ;CPU ERROR
3904 015206 022706 125252      CMP      #125252,R6          ;SET USER STACK TO ALTERNATING PATTERN
3905 015212 001401          BEQ      7$          ;IS USER SP CORRECT
3906                                     ;YES GO ON
3907 015214 104001          ERROR    +1          ;NO GO TO ERROR
3908 015216 012706 052525      7$:  MOV      #52525,R6          ;CPU ERROR
3909 015222 022706 052525      CMP      #52525,R6          ;SET USER STACK TO ALTERNATING PATTERN
3910 015226 001401          BEQ      8$          ;IS USER SP CORRECT
3911                                     ;YES GO ON
3912 015230 104001          ERROR    +1          ;NO GO TO ERROR
3913 015232 012706 000600      8$:  MOV      #600,R6          ;CPU ERROR
3914 015236 005037 177776      CLR      #177776          ;SETUP USER SP
3915 015242 012706 001000      MOV      #STBOT,R6          ;SET PS TO KER MODE
3916 015246 005037 000700      CLR      #700          ;SETUP KERNEL SP
3917 015252 005037 000600      CLR      #600          ;CLEAR FIRST WORDS OF SUP, KER, AND USE STACKS

```

***** DOUBLE OPERAND TESTS *****

```

3918 015256 005037 001000          CLR      @STBOT
3919 015262 004767 000054          JSR      PC,CHECK          ; TEST KER, SUP, AND USE STACKS
3920 015266 012737 040000 177776 RET1:  MOV      @40000,@177776    ;SET PSW TO SUP MODE
3921 015274 022706 000700          CMP      @700,R6          ;IS SUP SP CORRECT
3922 015300 001401          BEQ      1$              ;YES GO ON
3923                                ;NO GO TO ERROR
3924 015302 104001          ERROR   +1              ;CPU ERROR
3925 015304 012737 140000 177776 1$:  MOV      @140000,@177776    ;SET PSW TO USE MODE
3926 015312 022706 000600          CMP      @600,R6          ;IS USE SP CORRECT
3927 015316 001401          BEQ      2$              ;YES GO ON
3928                                ;NO GO TO ERROR
3929 015320 104001          ERROR   +1              ;CPU ERROR
3930 015322 005037 177776          2$:  CLR      @177776          ;SET PSW TO KER MODE
3931 015326 022706 001000          CMP      @STEOT,R6        ;IS KER SP CORRECT
3932 015332 001401          BEQ      3$              ;YES GO ON
3933                                ;NO GO TO ERROR
3934 015334 104001          ERROR   +1              ;CPU ERROR
3935 015336          3$:
3936 015336 000167 000206          JMP      MTS0
3937                                ;
3938                                ;ROUTINE TO CHECK STACKS AFTER TWO RTS STATEMENTS
3939                                ;
3940 015342 012737 040000 177776 CHECK:  MOV      @40000,@177776    ;SET PSW TO SUP MODE
3941 015350 004767 000044          JSR      PC,CHECK1        ; TEST SUP, KER, AND USE STACKS
3942 015354 022716 000000          RET2:  CMP      @0,(SP)      ;IS SUP STACK CLEARED
3943 015360 001401          BEQ      1$              ;YES GO ON
3944                                ;NO GO TO ERROR
3945 015362 104001          ERROR   +1              ;CPU ERROR
3946 015364 012737 140000 177776 1$:  MOV      @140000,@177776    ;SET PSW TO USE MODE
3947 015372 022716 000000          CMP      @0,(SP)          ;IS USE STACK CLEARED
3948 015376 001401          BEQ      2$              ;YES GO ON
3949                                ;NO GO TO ERROR
3950 015400 104001          ERROR   +1              ;CPU ERROR
3951 015402 005037 177776          2$:  CLR      @177776          ;SET PSW TO KER MODE
3952 015406 022716 015266          CMP      @RET1,(SP)       ;DOES KER STACK HAVE CORRECT DATA
3953 015412 001401          BEQ      3$              ;YES GO ON
3954                                ;NO GO TO ERROR
3955 015414 104001          ERROR   +1              ;CPU ERROR
3956 015416 000207          3$:  RTS      PC              ;RETURN
3957                                ;
3958                                ;ROUTINE TO CHECK STACKS AFTER ONE RTS
3959                                ;
3960 015420 012737 140000 177776 CHECK1:  MOV      @140000,@177776    ;SET PSW TO USE MODE
3961 015426 004767 000044          JSR      PC,CHECK2        ; TEST KER, SUP, AND USE STACKS
3962 015432 022716 000000          RET3:  CMP      @0,(SP)      ;IS USE STACK CLEARED
3963 015436 001401          BEQ      1$              ;YES GO ON
3964                                ;NO GO TO ERROR
3965 015440 104001          ERROR   +1              ;CPU ERROR
3966 015442 005037 177776          1$:  CLR      @177776          ;SET PSW TO KER MODE
3967 015446 022716 015266          CMP      @RET1,(SP)       ;IS KER STACK CORRECT
3968 015452 001401          BEQ      2$              ;YES GO ON
3969                                ;NO GO TO ERROR
3970 015454 104001          ERROR   +1              ;CPU ERROR
3971 015456 012737 040000 177776 2$:  MOV      @40000,@177776    ;SET PSW TO SUP MODE
3972 015464 022716 015354          CMP      @RET2,(SP)       ;IS SUP STACK CORRECT
3973 015470 001401          BEQ      3$              ;YES GO ON
3974                                ;NO GO TO ERROR

```

***** DOUBLE OPERAND TESTS *****

```

3975 015472 104001          ERROR +1          ;CPU ERROR
3976 015474 000207        3$: RTS PC          ;RETURN
3977
3978          ;
3979          ;ROUTINE TO CHECK STACKS AFTER ZERO RTS
3980 015476 022716 015432 CHECK2: CMP 0RET3,(SP)      ;IS USE STACK CORRECT
3981 015502 001401        BEQ 1$          ;YES GO ON
3982          ;NO GO TO ERROR
3983 015504 104001          ERROR +1          ;CPU ERROR
3984 015506 012737 040000 177776 1$: MOV 040000,00177776 ;SET PSW TO SUP MODE
3985 015514 022716 015354        CMP 0RET2,(SP)      ;IS SUP STACK CORRECT
3986 015520 001401        BEQ 2$          ;YES GO ON
3987          ;NO GO TO ERROR
3988 015522 104001          ERROR +1          ;CPU ERROR
3989 015524 005037 177776        CLR 00177776        ;SET PSW TO KER MODE
3990 015530 022716 015266        CMP 0RET1,(SP)      ;IS KER STACK CORRECT
3991 015534 001401        BEQ 3$          ;YES GO ON
3992          ;NO GO TO ERROR
3993 015536 104001          ERROR +1          ;CPU ERROR
3994 015540 012737 140000 177776 3$: MOV 0140000,00177776 ;SET PSW TO USE MODE
3995 015546 000207        RTS PC          ;RETURN
3996
3997 015550          ;
4000 015550          MTSO:
4001          ;
4002          ;
4003 015550 000277          ;
4004 015552 000244          CLZ          ;CC=1011
4005 015554 012704 000000        MOV 00,R4          ;CC=0101, R4=0
4006 015560 100403          BMI 1$          ;ERROR IF N FLAG
4007 015562 102402          BVS 1$          ;ERROR IF V FLAG SET
4008 015564 103001          BCC 1$          ;ERROR IF C FLAG CLEAR
4009 015566 001401          BEQ 2$          ;SKIP IF Z FLAG SET
4010          ;CC SHOULD=0101
4011 015570 104001        1$: ERROR +1          ;CPU ERROR
4012 015572 000277        2$: SCC          ;
4013 015574 000251          .WORD 251          ;CC=0110
4014 015576 012704 100000        MOV 0100000,R4      ;R4=100000, CC=1000
4015 015602 001403          BEQ 3$          ;ERROR IF Z SET
4016 015604 102402          BVS 3$          ;ERROR IF V SET
4017 015606 103401          BCS 3$          ;ERROR IF C SET
4018 015610 100401          BMI 4$          ;EXIT IF N SET
4019 015612 104001        3$: ERROR +1          ;CPU ERROR
4020          ;CC SHOULD= 1000
4021 015614        4$:
4022
4023
4026 015614          MBTCC:
4027
4028          ;
4029 015614 005004          ;
4030 015616 005104          CLR R4          ;R4=-1
4031 015620 000277          COM R4          ;
4032 015622 000244          SCC          ;
4033 015624 032704 000000        CLZ          ;CC=1011
4034 015630 100403          BIT 00,R4        ;CC=0101
4035 015632 102402          BMI 1$          ;ERROR IF N FLAG
          BVS 1$          ;ERROR IF V FLAG SET

```

***** DOUBLE OPERAND TESTS *****

```

4036 015634 103001          BCC      1#           ;ERROR IF C FLAG CLEAR
4037 015636 001401          BEQ      2#           ;SKIP IF Z FLAG SET
4038 015640 104001          1#:     ERROR      +1       ;CPU ERROR
4039                                ;CC SHOULD=0101
4040 015642 000277          2#:     SCC
4041 015644 000251          .WORD   251           ;CC=0110
4042 015646 032704 100000  BIT      @100000,R4    ;CC=1000
4043 015652 001403          BEQ      3#           ;ERROR IF Z SET
4044 015654 102402          BVS      3#           ;ERROR IF V SET
4045 015656 103401          BCS      3#           ;ERROR IF C SET
4046 015660 100401          BMI      4#           ;EXIT IF N SET
4047 015662 104001          3#:     ERROR      +1       ;CPU ERROR
4048                                ;CC SHOULD= 1000
4049 015664          4#:
4050
4051
4054 015664          MBCCC:
4055
4056          ;      TEST BIC CONDITION CODES - **0-
4057 015664 005004          CLR      R4
4058 015666 005104          COM      R4           ;R4=-1
4059 015670 000277          SCC
4060 015672 000244          CLZ
4061 015674 042704 177777  BIC      @177777,R4    ;CC=1011
4062 015700 100403          BMI      1#           ;CC=0101
4063 015702 102402          BVS      1#           ;ERROR IF N FLAG
4064 015704 103001          BVS      1#           ;ERROR IF V FLAG SET
4065 015706 001401          BCC      1#           ;ERROR IF C FLAG CLEAR
4066 015710 104001          BEQ      2#           ;SKIP IF Z FLAG SET
4067          1#:     ERROR      +1       ;CPU ERROR
4068          2#:     COM      R4           ;CC SHOULD=0101
4069          SCC           ;R4=-1
4070 015716 000251          .WORD   251           ;CC=0110
4071 015720 042704 077777  BIC      @77777,R4    ;CC=1000
4072 015724 001403          BEQ      3#           ;ERROR IF Z SET
4073 015726 102402          BVS      3#           ;ERROR IF V SET
4074 015730 103401          BCS      3#           ;ERROR IF C SET
4075 015732 100401          BMI      4#           ;EXIT IF N SET
4076 015734 104001          3#:     ERROR      +1       ;CPU ERROR
4077                                ;CC SHOULD= 1000
4078 015736          4#:
4079
4080
4083 015736          MBSCC:
4084
4085          ;      TEST BIS CONDITION CODES
4086 015736 005004          CLR      R4           ;R4=0
4087 015740 000277          SCC
4088 015742 000246          .WORD   246           ;CC=1001
4089 015744 052704 000000  BIS      @0,R4        ;R4=0, CC=0101
4090 015750 100403          BMI      1#           ;ERROR IF MINUS
4091 015752 102402          BVS      1#           ;ERROR IF V SET
4092 015754 103001          BCC      1#           ;ERROR IF C CLEAR
4093 015756 001401          BEQ      2#           ;BRANCH IF GOOD
4094 015760 104001          1#:     ERROR      +1       ;CPU ERROR
4095                                ;BIS CC FAILED
4096 015762 000277          2#:     SCC

```


***** DOUBLE OPERAND TESTS *****

```

4097 015764 000241          CLC                      ;CC=1110
4098 015766 052704 100076  BIS      #100076,R4      ;R4=100076, CC=1000
4099 015772 001403          BEQ      3#              ;ERROR IF Z SET
4100 015774 102402          BVS      3#              ;ERROR IF V SET
4101 015776 103401          BCS      3#              ;ERROR IF C SET
4102 016000 100401          BMI      4#              ;BRANCH IF GOOD
4103 016002 104001          3#:  ERROR      +1      ;CPU ERROR
4104                                ;BAD BIS CC
4105 016004          4#:
4106
4107
4110 016004          MDCCC:
4111
4112          ;      TEST DEC, INC CONDITION CODES
4113 016004 012704 077777  MOV      #77777,R4      ;R4=77777
4114 016010 000257          CCC
4115 016012 000261          SEC                      ;CC=0001
4116 016014 005204          INC      R4              ;R4=100000, CC=0011
4117 016016 001403          BEQ      1#              ;ERROR IF ZERO
4118 016020 100002          BPL      1#              ;ERROR IF POSITIVE
4119 016022 102001          BVC      1#              ;ERROR IF V CLEAR
4120 016024 103401          BCS      2#              ;BRANCH IF GOOD
4121 016026 104001          1#:  ERROR      +1      ;CPU ERROR
4122                                ;INC FAILED
4123 016030 000257          2#:  CCC
4124 016032 005204          INC      R4              ;R4=100001, CC=1000
4125 016034 103413          BCS      3#              ;ERROR IF C SET
4126 016036 102412          BVS      3#              ;ERROR IF V SET
4127 016040 005304          DEC      R4              ;R4=100000, CC=1000
4128 016042 102410          BVS      3#              ;ERROR IF V SET
4129 016044 103407          BCS      3#              ;ERROR IF C SET
4130 016046 000277          SCC
4131 016050 000252          .WORD   252              ;CC=0101
4132 016052 005304          DEC      R4              ;R4=77777, CC=1011
4133 016054 001403          BEQ      3#              ;ERROR IF Z SET
4134 016056 102002          BVC      3#              ;ERROR IF V CLEAR
4135 016060 103C01          BCC      3#              ;ERROR IF C CLEAR
4136 016062 100001          BPL      4#              ;BRANCH IF GOOD
4137 016064 104001          3#:  ERROR      +1      ;CPU ERROR
4138                                ;BAD CC
4139 016066          4#:
4140
4141
4151 016066          MCTSCC:
4152
4153          ;      TEST CLR, TST, SWAB CONDITION CODES
4154          ;*****
          ;0100 - **00 - **00
4155 016066 000277          SCC
4156 016070 000244          CLZ                      ;CC=1011
4157 016072 005004          CLR      R4              ;R4=0, CC=0100
4158 016074 100403          BMI      1#              ;ERROR IF MINUS
4159 016076 102402          BVS      1#              ;ERROR IF V SET
4160 016100 103401          BCS      1#              ;ERROR IF C SET
4161 016102 001401          BEQ      2#              ;BRANCH IF GOOD
4162 016104 104001          1#:  ERROR      +1      ;CPU ERROR
4163                                ;BAD CC ON CLR

```

***** DOUBLE OPERAND TESTS *****

```

4164 016106 005104      2$:  COM      R4              ;R4=-1
4165 016110 000277      SCC              ;CC=1111
4166 016112 005704      TST      R4              ;CC=1000
4167 016114 001403      BEQ      3$              ;ERROR IF Z SET
4168 016116 102402      BVS      3$              ;ERROR IF V SET
4169 016120 103401      BCS      3$              ;ERROR IF C SET
4170 016122 100401      BMI      4$              ;BRANCH IF GOOD
4171 016124 104001      3$:  ERROR      +1      ;CPU ERROR
4172                                     ;BAD TST CC
4173 016126 000277      4$:  SCC              ;CC=1111
4174 016130 000304      SWAB     R4              ;CC=1000
4175 016132 102402      BVS      5$              ;ERROR IF V SET
4176 016134 103401      BCS      5$              ;ERROR IF C SET
4177 016136 100401      BMI      6$              ;BRANCH IF GOOD
4178 016140 104001      5$:  ERROR      +1      ;CPU ERROR
4179                                     ;BAD SWAB CC
4180 016142
4181
4182
4185 016142      MADCC:
4186
4187      ;      TEST ADD CONDITION CODES
4188 016142 012704 077777      MOV      #77777,R4      ;R4=77777
4189 016146 012701 000001      MOV      #1,R1          ;R1=1
4190 016152 000257      CCC              ;CC=0000
4191 016154 060401      ADD      R4,R1          ;77777 + 1 = 100000 IN R1
4192 016156 102003      BVC      1$              ;ERROR IF V CLEAR
4193 016160 103402      BCS      1$              ;ERROR IF CARRY
4194 016162 001401      BEQ      1$              ;ERROR IF Z SET
4195 016164 100401      BMI      2$              ;BRANCH IF GOOD
4196 016166 104001      1$:  ERROR      +1      ;CPU ERROR
4197                                     ;CC SHOULD = 1010
4198 016170 005204      2$:  INC      R4          ;R4=100000
4199 016172 060401      ADD      R4,R1          ;100000 + 100000 = 0 IN R1
4200 016174 102002      BVC      3$              ;ERROR IF V CLEAR
4201 016176 103001      BCC      3$              ;ERROR IF CARRY CLEAR
4202 016200 001401      BEQ      4$              ;BRANCH IF GOOD
4203 016202 104001      3$:  ERROR      +1      ;CPU ERROR
4204                                     ;CC SHOULD = 0111
4205 016204 060401      4$:  ADD      R4,R1          ;0 + 100000 = 100000
4206 016206 102402      BVS      5$              ;ERROR IF V SET
4207 016210 103401      BCS      5$              ;ERROR IF SET
4208 016212 100401      BMI      5$              ;BRANCH IF GOOD
4209 016214 104001      5$:  ERROR      +1      ;CPU ERROR
4210                                     ;CC SHOULD = 1000
4211 016216
4212
4213
4216 016216      MACCC:
4217
4218      ;      TEST ADC CONDITION CODES
4219 016216 012704 177777      MOV      #177777,R4     ;R4=177777
4220 016222 000277      SCC              ;CC=1111
4221 016224 005504      ADC      R4              ;R4=0 CC=0101
4222 016226 100403      BMI      1$              ;ERROR IF MINUS
4223 016230 102402      BVS      1$              ;ERROR IF V SET
4224 016232 103001      BCC      1$              ;ERROR IF C SET

```

***** DOUBLE OPERAND TESTS *****

```

4225 016234 001401          BEQ      2$
4226 016236 104001          1$:   ERROR    +1          ;CPU ERROR
4227                                ;BAD ADC
4228 016240 012704 077777  2$:   MOV      #077777,R4          ;R4=077777
4229 016244 000277          SCC
4230 016246 000242          CLV          ;CC=1101
4231 016250 005504          ADC      R4          ;R4=100000 CC=1010
4232 016252 100003          BPL      3$          ;ERROR IF PLUS
4233 016254 103402          BCS      3$          ;ERROR IF C SET
4234 016256 001401          BEQ      3$          ;ERROR IF ZERO
4235 016260 102401          BVS      4$          ;BRANCH IF GOOD
4236 016262 104001          3$:   ERROR    +1          ;CPU ERROR
4237                                ;BAD ADC
4238 016264 000277          4$:   SCC
4239 016266 005504          ADC      R4          ;CC=1111
4240 016270 102402          BVS      5$          ;R4=100000 CC=1000
4241 016272 103401          BCS      5$          ;ERROR IF V SET
4242 016274 100401          BMI      6$          ;ERROR IF C SET
4243 016276 104001          5$:   ERROR    +1          ;BRANCH IF GOOD
4244                                ;CPU ERROR
4245 016300          6$:                                ;BAD ADC CC SHOULD= 1000
4246
4247
4250 016300          MNCCCC:
4251
4252          ; TEST NEG, CMP, COM CONDITION CODES
4253 016300 012704 077777  MOV      #077777,R4          ;R4=77777
4254 016304 000257          CCC          ;CC=0000
4255 016306 005404          NEG      R4          ;R4=100001 CC=1001
4256 016310 102403          BVS      1$          ;ERROR IF V SET
4257 016312 103002          BCC      1$          ;ERROR IF C CLEAR
4258 016314 001401          BEQ      1$          ;ERROR IF Z SET
4259 016316 100401          BMI      2$          ;BRANCH IF GOOD
4260 016320 104001          1$:   ERROR    +1          ;CPU ERROR
4261                                ;BAD NEGATE
4262 016322 005004          2$:   CLR      R4          ;R4=0
4263 016324 000257          CCC          ;CC=0000
4264 016326 005404          NEG      R4          ;CC=0101
4265 016330 100403          BMI      3$          ;ERROR IF N SET
4266 016332 103402          BCS      3$          ;ERROR IF C SET
4267 016334 102401          BVS      3$          ;ERROR IF V SET
4268 016336 001401          BEQ      4$          ;BRANCH IF GOOD
4269 016340 104001          3$:   ERROR    +1          ;CPU ERROR
4270                                ;BAD NEG
4271 016342 012704 077777  4$:   MOV      #77777,R4          ;R4=77777
4272 016346 012701 170000  MOV      #170000,R1          ;R1=170000
4273 016352 000257          CCC          ;CC=0000
4274 016354 020401          CMP      R4,R1          ;77777 170000 = 107777 CC= 1011
4275 016356 102003          BVC      5$          ;ERROR IF V CLEAR
4276 016360 103002          BCC      5$          ;ERROR IF C CLEAR
4277 016362 001401          BEQ      5$          ;ERROR IF ZERO
4278 016364 100401          BMI      6$          ;BRANCH IF GOOD
4279 016366 104001          5$:   ERROR    +1          ;CPU ERROR
4280                                ;BAD CMP
4281 016370 000257          6$:   CCC
4282 016372 005101          COM      R1          ;R1=7777
4283 016374 100403          BMI      7$          ;ERROR IF MINUS

```

***** DOUBLE OPERAND TESTS *****

```

4284 016376 001402      BEQ      7$      ;ERROR IF ZERO
4285 016400 103001      BCC      7$      ;ERROR IF CARRY
4286 016402 102001      BVC      8$      ;BRANCH IF GOOD
4287 016404 104001      7$:  ERROR    +1      ;CPU ERROR
4288                                ;BAD COM
4289 016406 000277      8$:  SCC
4290 016410 005101      COM      R1
4291 016412 100401      BMI     10$     ;BRANCH IF GOOD
4292 016414 104001      9$:  ERROR    +1      ;CPU ERROR
4293                                ;BAD COM
4294 016416      10$:
4295
4296
4299 016416      MSBCC:
4300
4301      ;      TEST SUB CONDITION CODES
4302 016416 012704 077775      MOV      #77775,R4      ;R4=77775
4303 016422 000257      CCC      ;CC=0000
4304 016424 162704 137757      SUB      #137757,R4     ;77775 - 137757
4305                                ;TRY TO CAUSE AN ARITHMETIC OVERFLOW
4306 016430 102003      BVC      1$      ;ERROR IF V CLEAR
4307 016432 100002      BPL      1$      ;ERROR IF RESULT IS POSITIVE
4308 016434 001401      BEQ      1$      ;ERROR IF Z SET
4309 016436 103401      BCS      2$      ;BRANCH IF GOOD
4310 016440 104001      1$:  ERROR    +1      ;CPU ERROR
4311                                ;BAD SUBTRACT
4312 016442 012704 000005      2$:  MOV      #5,R4      ;R4=5
4313 016446 000257      CCC      ;CC=0000
4314 016450 162704 000012      SUB      #12,R4      ;5-12=-5 AND SETS CARRY
4315 016454 103003      BCC      3$      ;ERROR IF CARRY CLEAR
4316 016456 102402      BVS      3$      ;ERROR IF OVERFLOW
4317 016460 001401      BEQ      3$      ;ERROR IF ZERO
4318 016462 100401      BMI     4$      ;BRANCH IF GOOD
4319 016464 104001      3$:  ERROR    +1      ;CPU ERROR
4320                                ;SUBTRACT FAILED
4321 016466      4$:
4322
4323
4326 016466      MSBCCC:
4327
4328      ;      TEST SBC CONDITION CODES
4329 016466 012704 100000      MOV      #100000,R4     ;R4=100000
4330 016472 000257      CCC      ;C=0000
4331 016474 005604      SBC      R4      ;TRY TO SET V
4332 016476 100006      BPL      1$      ;ERROR IF N CLEAR
4333 016500 102405      BVS      1$      ;ERROR IF V SET (HAVENT SET C YET)
4334 016502 000261      SEC      ;CC SHOULD = 1001
4335 016504 005604      SBC      R4      ;TRY AGAIN TO SET V
4336 016506 102002      BVC      1$      ;ERROR IF V CLEAR
4337 016510 103401      BCS      1$      ;ERROR IF C SET
4338 016512 100001      BPL      2$      ;BRANCH IF GOOD
4339 016514 104001      1$:  ERROR    +1      ;CPU ERROR
4340                                ;SBC FAILED
4341 016516 005004      2$:  CLR      R4      ;R4=0
4342 016520 000277      SCC
4343 016522 000241      CLC      ;CC=1110
4344 016524 005604      SBC      R4      ;TRY TO CAUSE C FLAG FAILURE

```

***** DOUBLE OPERAND TESTS *****

```

4345 016526 103410          BCS      3$          ;ERROR IF C SET
4346 016530 102407          BVS      3$          ;ERROR IF V SET
4347 016532 001006          BNE      3$          ;ERROR IF NOT ZERO
4348 016534 000261          SEC              ;SET CARRY
4349 016536 005604          SBC      R4          ;NOW, 0 - CARRY = 177777
4350 016540 103003          BCC      3$          ;ERROR IF CARRY CLEAR
4351 016542 102402          BVS      3$          ;ERROR IF V SET
4352 016544 001401          BEQ      3$          ;ERROR IF ZERO
4353 016546 100401          BMI      4$          ;BRANCH IF GOOD
4354 016550 104001          3$:      ERROR      +1          ;CPU ERROR
4355                                     ;SBC FAILED
4356 016552          4$:
4357
4358
4361 016552          MRLCC:
4362
4363          ;      TEST ROL CONDITION CODES
4364 016552 012704 060000      MOV      #60000,R4          ;R4= 0110000000000000
4365 016556 000257          CCC              ;CC=0000
4366 016560 006104          ROL      R4          ;R4= 1100000000000000
4367 016562 103402          BCS      1$          ;ERROR IF CARRY
4368 016564 102001          BVC      1$          ;ERROR IF V CLEAR
4369 016566 100401          BMI      2$          ;BRANCH IF GOOD
4370 016570 104001          1$:      ERROR      +1          ;CPU ERROR
4371                                     ;ROL FAILED
4372 016572 006104          2$:      ROL      R4          ;R4= 1000000000000000
4373 016574 103002          BCC      3$          ;ERROR IF CARRY CLEAR
4374 016576 102401          BVS      3$          ;ERROR IF V SET
4375 016600 100401          BMI      4$          ;BRANCH IF GOOD
4376 016602 104001          3$:      ERROR      +1          ;CPU ERROR
4377                                     ;BAD ROL
4378 016604 006104          4$:      ROL      R4          ;R4 = 0000000000000001
4379 016606 102003          BVC      5$          ;ERROR IF V CLEAR
4380 016610 103002          BCC      5$          ;ERROR IF C CLEAR
4381 016612 100401          BMI      5$          ;ERROR IF MINUS
4382 016614 001001          BNE      6$          ;BRANCH IF GOOD
4383 016616 104001          5$:      ERROR      +1          ;CPU ERROR
4384                                     ;BAD ROL
4385 016620 006104          6$:      ROL      R4          ;R4=0000000000000011
4386 016622 102402          BVS      7$          ;ERROR IF V SET
4387 016624 103401          BCS      7$          ;ERROR IF C SET
4388 016626 100001          BPL      8$          ;BRANCH IF GOOD
4389 016630 104001          7$:      ERROR      +1          ;CPU ERROR
4390                                     ;BAD ROL
4391 016632          8$:
4392
4393
4396 016632          MRRCC:
4397
4398          ;      TEST ROR CONDITION CODES
4399 016632 012704 000003      MOV      #3,R4          ;R4= 0000000000000011
4400 016636 000257          CCC              ;CC= 0000
4401 016640 006004          ROR      R4          ;R4= 0000000000000001
4402 016642 103002          BCC      1$          ;ERROR IF NO CARRY
4403 016644 102001          BVC      1$          ;ERROR IF V CLEAR
4404 016646 100001          BPL      2$          ;BRANCH IF GOOD
4405 016650 104001          1$:      ERROR      +1          ;CPU ERROR

```

***** DOUBLE OPERAND TESTS *****

```

4406                                     ;ROR FAILED
4407 016652 006004 2$: ROR R4           ;R4= 1000000000000000
4408 016654 103002       BCC 3$       ;ERROR IF CARRY CLEAR
4409 016656 102401       BVS 3$       ;ERROR IF V SET
4410 016660 100401       BMI 4$       ;BRANCH IF GOOD
4411 016662 104001 3$: ERROR +1       ;CPU ERROR
4412                                     ;BAD ROR
4413 016664 006004 4$: ROR R4           ;R4 = 1100000000000000
4414 016666 102002       BVC 5$       ;ERROR IF V
4415 016670 103401       BCS 5$       ;ERROR IF C SET
4416 016672 100401       BMI 6$       ;BRANCH IF GOOD
4417 016674 104001 5$: ERROR +1       ;CPU ERROR
4418                                     ;BAD ROR
4419 016676 006004 6$: ROR R4           ;R4= 0110000000000000
4420 016700 102402       BVS 7$       ;ERROR IF V SET
4421 016702 103401       BCS 7$       ;ERROR IF C SET
4422 016704 100001       BPL 8$       ;BRANCH IF GOOD
4423 016706 104001 7$: ERROR +1       ;CPU ERROR
4424                                     ;BAD ROR
4425 016710 8$:
4426
4427
4430 016710 XCBIT:
4439
4440 ; TEST C BIT WITH ROR/ROL
4441 ;:*****
;THIS TEST IS TO CHECK FOR A SLOW C BIT PATH INTERNAL TO THE J11 DATA CHIP
;PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 AND 1593)
4442 016710 012701 052525 MOV #52525,R1 ;INIT R1 WITH DATA
4443 016714 000241 CLC ;CLEAR THE C BIT
4444 016716 006001 ROR R1 ;R1=025252, C BIT =1
4445 016720 006001 ROR R1 ;R1=112525, C BIT =0
4446 016722 006001 ROR R1 ;R1=045252, C BIT =1
4447 016724 103401 BCS 1$ ;BRANCH IF CARRY BIT SET
4448 016726 104001 ERROR +1 ;CPU ERROR
4449 016730 022701 045252 1$: CMP #45252,R1 ;IS DATA IN R1 = TO EXPECTED DATA?
4450 016734 001401 BEQ 2$ ;BRANCH IF YES
4451 016736 104001 ERROR +1 ;CPU ERROR
4452 016740 012701 125252 2$: MOV #125252,R1 ;SET UP R1
4453 016744 000241 CLC ;CLEAR THE CARRY BIT
4454 016746 006101 ROL R1 ;R1=052524, C BIT =1
4455 016750 006101 ROL R1 ;R1=125251, C BIT =0
4456 016752 006101 ROL R1 ;R1=052522, C BIT =1
4457 016754 103401 BCS 3$ ;BRANCH IF CARRY SET
4458 016756 104001 ERROR +1 ;CPU ERROR
4459 016760 022701 052522 3$: CMP #052522, R1 ;IS DATA IN R1 = TO EXPECTED DATA?
4460 016764 001401 BEQ 4$
4461 016766 104001 ERROR +1 ;CPU ERROR
4462 016770 4$:
4464
4466 016770 MALCC:
4467
4468 ; TEST ASL CONDITION CODES
4469 016770 012704 060000 MOV #60000,R4 ;R4= 0110000000000000
4470 016774 000257 CCC ;CC=0000
4471 016776 006304 ASL R4 ;C=0 R4= 1100000000000000
4472 017000 103402 BCS 1$ ;ERROR IF CARRY

```

***** DOUBLE OPERAND TESTS *****

```

4473 017002 102001          BVC      1$          ;ERROR IF V CLEAR
4474 017004 100401          BMI      2$          ;BRANCH IF GOOD
4475 017006 104001          1$:      ERROR    +1          ;CPU ERROR
4476                                ;ASL FAILED
4477 017010 006304          2$:      ASL      R4          ;C=1 R4= 1000000000000000
4478 017012 103002          BCC      3$          ;ERROR IF CARRY CLEAR
4479 017014 102401          BVS      3$          ;ERROR IF V SET
4480 017016 100401          BMI      4$          ;BRANCH IF GOOD
4481 017020 104001          3$:      ERROR    +1          ;CPU ERROR
4482                                ;BAD ASL
4483 017022 006304          4$:      ASL      R4          ;C=1 R4= 0000000000000000
4484 017024 102003          BVC      5$          ;ERROR IF V CLEAR
4485 017026 103002          BCC      5$          ;ERROR IF C CLEAR
4486 017030 100401          BMI      5$          ;ERROR IF MINUS
4487 017032 001401          BEQ      6$          ;BRANCH IF GOOD
4488 017034 104001          5$:      ERROR    +1          ;CPU ERROR
4489                                ;BAD ASL
4490 017036 006304          6$:      ASL      R4          ;C=0 R4= 0000000000000000
4491 017040 102402          BVS      7$          ;ERROR IF V SET
4492 017042 103401          BCS      7$          ;ERROR IF C SET
4493 017044 100001          BPL      8$          ;BRANCH IF GOOD
4494 017046 104001          7$:      ERROR    +1          ;CPU ERROR
4495                                ;BAD ASL
4496 017050          8$:
4497
4500 017050          MARCC:
4501
4502          ;      TEST ASR CONDITION CODES
4503 017050 012704 000341      MOV      #341,R4          ;R4= 0000000011100001
4504 017054 000257          CCC          ;CC=0000
4505 017056 006204          ASR      R4          ;R4= 0000000001110000
4506 017060 103002          BCC      1$          ;ERROR IF NO CARRY
4507 017062 102001          BVC      1$          ;ERROR IF V CLEAR
4508 017064 100001          BPL      2$          ;BRANCH IF GOOD
4509 017066 104001          1$:      ERROR    +1          ;CPU ERROR
4510                                ;ASR FAILED
4511 017070 052704 100001      2$:      BIS      #100001,R4          ;R4= 1000000001110001
4512 017074 006204          ASR      R4          ;R4= 1100000000111000
4513 017076 103002          BCC      3$          ;ERROR IF CARRY CLEAR
4514 017100 102401          BVS      3$          ;ERROR IF V SET
4515 017102 100401          BMI      4$          ;BRANCH IF GOOD
4516 017104 104001          3$:      ERROR    +1          ;CPU ERROR
4517                                ;BAD ASR
4518 017106 006204          4$:      ASR      R4          ;R4= 1110000000011100
4519 017110 102002          BVC      5$          ;ERROR IF V
4520 017112 103401          BCS      5$          ;ERROR IF C SET
4521 017114 100401          BMI      6$          ;BRANCH IF GOOD
4522 017116 104001          5$:      ERROR    +1          ;CPU ERROR
4523                                ;BAD ASR
4524 017120 006204          6$:      ASR      R4          ;R4= 1111000000001110
4525 017122 102005          VC      7$          ;ERROR IF V CLEAR
4526 017124 103404          JCS      7$          ;ERROR IF C SET
4527 017126 100003          JPL      7$          ;ERROR IF PLUS
4528 017130 022704 170016      CMP      #170016,R4          ;SEE IF EXPECTED RESULT
4529 017134 001401          BEQ      8$          ;BRANCH IF GOOD
4530 017136 104001          7$:      ERROR    +1          ;CPU ERROR
4531                                ;BAD ASR

```


***** DOUBLE OPERAND TESTS *****

```

452 017140      8$:
4533
4534
4537 017140      MSXTCC:
4538
4539      ;      TEST SXT CONDITION CODES / --*0-
4540 017140 012704 123456      MOV      #123456,R4      ;R4=123456
4541 017144 010401      MOV      R4,R1      ;SAVE CONTENTS
4542 017146 000257      CCC      ;CC=0000
4543 017150 006704      SXT      R4      ;R4=0 CC=0100
4544 017152 103403      BCS      1$      ;ERROR IF CARRY
4545 017154 100402      BMI      1$      ;ERROR IF MINUS
4546 017156 102401      BVS      1$      ;ERROR IF OVERFLOW
4547 017160 001401      BEQ      2$      ;BRANCH IF GOOD
4548 017162 104001      1$:      ERROR      +1      ;CPU ERROR
4549      ;BAD SXT
4550 017164 010104      2$:      MOV      R1,R4      ;RESTORE R4
4551 017166 000277      SCC      ;CC=1111
4552 017170 006704      SXT      R4      ;R4= 1 CC=1001
4553 017172 001405      BEQ      3$      ;ERROR IF ZERO
4554 017174 100004      BPL      3$      ;ERROR IF PLUS
4555 017176 105003      BCC      3$      ;ERROR IF NO CARRY
4556 017200 102402      BVS      3$      ;ERROR IF OVERFLOW
4557 017202 005104      COM      R4      ;R4=0
4558 017204 001401      BEQ      4$      ;BRANCH IF GOOD
4559 017206 104001      3$:      ERROR      +1      ;CPU ERROR
4560      ;BAD SXT
4561 017210      4$:
4562
4563
4566 017210      MXRCC:
4567
4568      ;      TEST XOR CONDITION CODES / **0-
4569 017210 012704 123456      MOV      #123456,R4      ;R4=123456
4570 017214 012701 052525      MOV      #52525,R1      ;R1=52525
4571 017220 000257      CCC      ;CC=0000
4572 017222 074104      XOR      R1,R4      ;*TI* R4=171173
4573 017224 102403      BVS      1$      ;ERROR IF OVERFLOW
4574 017226 001402      BEQ      1$      ;ERROR IF ZERO
4575 017230 103401      BCS      1$      ;ERROR IF CARRY
4576 017232 100401      BMI      2$      ;BRANCH IF GOOD
4577 017234 104001      1$:      ERROR      +1      ;CPU ERROR
4578      ;BAD XOR
4579 017236 012701 125252      2$:      MOV      #125252,R1      ;R1=125252
4580 017242 000277      SCC      ;CC=1111
4581 017244 074104      XOR      R1,R4      ;R4=054321
4582 017246 100403      BMI      3$      ;ERROR IF MINUS
4583 017250 001402      BEQ      3$      ;ERROR IF ZERO
4584 017252 103001      BCC      3$      ;ERROR IF CARRY CLEAR
4585 017254 102001      BVC      4$      ;BRANCH IF GOOD
4586 017256 104001      3$:      ERROR      +1      ;CPU ERROR
4587      ;BAD XOR
4588 017260 074404      4$:      XOR      R4,R4      ;R4=0
4589 017262 102406      BVS      5$      ;ERROR IF OVREFLOW
4590 017264 100405      BMI      5$      ;ERROR IF MINUS
4591 017266 103004      BCC      5$      ;ERROR IF NO CARRY
4592 017270 001003      BNE      5$      ;ERROR IF NOT ZERO

```

***** DOUBLE OPERAND TESTS *****

```

4593 017272 022704 000000      CMP      #0,R4          ;SEE IF EXPECTED RESULT
4594 017276 001401              BEQ      6$            ;BRANCH IF GOOD
4595 017300 104001      5$:      ERROR      +1          ;CPU ERROR
4596                                ;BAD XOR
4597 017302      6$:
4598
4599
4600
4612 017302      ;
4613      MSXT:
4614      ;
4615      ;      TEST SXT (SIGN EXTEND INSTRUCTION)
;*****
;AN ADDITIONAL TEST IS INCLUDED TO CHECK FOR A SLOW N BIT PATH
;ON A TRANSITION FROM ZERO TO ONE INTERNAL TO THE J11 DATA CHIP
;THE PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 OR 1593)
4616 017302 005004      CLR      R4          ;TRASH R4
4617 017304 000257      CCC          ;CC=0000
4618 017306 000271      .WORD   271        ;CC=1001
4619 017310 006704      SXT      R4          ;*TEST INSTRUCTION
4620 017312 102405      BVS      1$          ;BRANCH IF OVERFLOW IS NOT CLEARED
4621 017314 100004      BPL      1$          ;BRANCH IF N BIT EFFECTED
4622 017316 001403      BEQ      1$          ;BRANCH IF Z BIT EFFECTED
4623 017320 103002      BCC      1$          ;BRANCH IF C BIT EFFECTED
4624 017322 005204      INC      R4
4625 017324 001401      BEQ      2$          ;BRANCH IF R4 =0
4626 017326 104001      1$:      ERROR      +1          ;CPU ERROR
4627                                ;CC SHOULD HAVE = 1101
4628 017330 000277      2$:      SCC
4629 017332 000250      CLN          ;CC=0111
4630 017334 005004      CLR      R4
4631 017336 012714 000055      MOV      #55,(R4)    ;TRASH R4
4632 017342 006714      SXT      (R4)        ;*TEST INSTRUCTION
4633 017344 001005      BNE      3$          ;BRANCH IF BIT EFFECTED
4634 017346 102404      BVS      3$          ;BRANCH IF OVERFLOW
4635 017350 103403      BCS      3$
4636 017352 100402      BMI      3$          ;BRANCH IF N IS SET
4637 017354 005714      TST      (R4)        ;VERIFY INSTRUCTION WORKED
4638 017356 001401      BEQ      4$          ;BRANCH IF R4=0
4639 017360 104001      3$:      ERROR      +1          ;CPU ERROR
4640                                ;SXT FAILED
4641
4642      ;      NOW TEST FOR SLOW N BIT IN J11 DATA CHIP
4643
4644 017362 012700 177777      4$:      MOV      #-1,R0      ;R0=177777, N BIT = 1
4645 017366 005004      CLR      R4          ;CLEAT THE N BIT
4646 017370 006700      SXT      R0          ;***TEST INSTRUCTION***
4647                                ;TEST N BIT TRANSITION 1 TO 0
4648 017372 005700      TST      R0          ;R0 SHOULD = 0
4649 017374 001401      BEQ      5$          ;BRANCH IF OK
4650 017376 104001      ERROR    +1          ;CPU ERROR
4651 017400 005000      5$:      CLR      R0          ;CLEAR R0, N BIT = 0
4652 017402 012704 177777      MOV      #-1,R4      ;SET N BIT
4653 017406 006700      SXT      R0          ;***TEST INSTRUCTION***
4654                                ;TEST N BIT TRANSITION 0 TO 1
4655 017410 022700 177777      CMP      #-1,R0      ;R0 SHOULD = 177777
4656 017414 001401      BEQ      6$          ;BRANCH IF OK
4657 017416 104001      ERROR    +1          ;CPU ERROR

```

***** DOUBLE OPERAND TESTS *****

```

4658 017420          6#:
4660                ;
4662 017420          MXOR:
4663
4664                ;
4665 017420 012701 007643      TEST XOR
4666 017424 012704 133333      MOV      #7643,R1          ;SETUP DATA
4667 017430 000277            MOV      #133333,R4       ;SETUP DATA
4668 017432 074401            SCC
4669 017434 100006            XOR      R4,R1           ;*TEST INSTRUCTION
4670 017436 001405            BPL      1#             ;BRANCH IF PLUS TO ERROR
4671 017440 103004            BEQ      1#             ;ERROR IF ZERO
4672 017442 102403            BCC      1#             ;ERROR IF CARRY CLEAR
4673 017444 020127 134570     BVS      1#             ;ERROR IF V SET
4674 017450 001401            CMP      R1,#134570     ;VERIFY CORRECT RESULT
4675 017452 104001            BEQ      2#             ;BRANCH IF GOOD
4676                1#:      ERROR      +1          ;CPU ERROR
4677 017454 010102            ;BAD XOR
4678 017456 000257            2#:      MOV      R1,R2
4679 017460 074402            CCC
4680 017462 100405            XOR      R4,R2           ;*TEST INSTRUCTION
4681 017464 102404            BMI      3#             ;ERROR IF MINUS
4682 017466 103403            BVS      3#             ;ERROR IF OVERFLOW
4683 017470 020227 007643     BCS      3#             ;ERROR IF CARRY
4684 017474 001401            CMP      R2,#7643
4685 017476 104001            BEQ      4#             ;BRANCH IF GOOD
4686                3#:      ERROR      +1          ;CPU ERROR
4687 017500            ;BAD XOR
4688                4#:
4690                ;
4692 017500          MSOB:
4693
4694                ;
4695 017500 012704 000555      TEST SOB
4696 017504 000277            MOV      #555,R4          ;SETUP TEST COUNTER
4697 017506 103015            SCC
4698 017510 102014            1#:      BCC      2#             ;CC=17
4699 017512 100013            BVC      2#             ;ERROR IF CARRY CLEAR
4700 017514 001012            BPL      2#             ;ERROR IF NO OVERFLOW
4701 017516 077405            BNE      2#             ;ERROR IF PLUS
4702 017520 103005            SOB      R4,1#         ;ERROR IF ZERO
4703 017522 102004            BCC      3#             ;*TEST INSTRUCTION
4704 017524 100003            BVC      3#             ;ERROR IF CARRY CLEAR
4705 017526 001002            BPL      3#             ;ERROR IF NO OVERFLOW
4706 017530 000167 000010     BNE      3#             ;ERROR IF PLUS
4707 017534 104001            JMP      4#             ;ERROR IF ZERO
4708                3#:      ERROR      +1          ;CPU ERROR
4709 017536 000167 000002      ;CC EFFECTED DURING TEST
4710 017542 104001            JMP      4#
4711                2#:      ERROR      +1          ;CPU ERROR
4712 017544 020427 000000     4#:      CMP      R4,#0       ;CC EFFECTED AFTER TEST
4713 017550 001401            BEQ      5#             ;IS R4 CORRECT
4714 017552 104001            ERROR    +1           ;YES GO ON
4715                ;CPU ERROR
4716 017554            5#:      ;NO GO TO ERROR
4717
4719

```

***** DOUBLE OPERAND TESTS *****

```

4721 017554          MARK:
4722
4723
4724 017554 012706 000700          ; MARK INSTRUCTION TEST
4725 017560 012737 125252 000776  MOV    #STBOT-100,SP          ;SETUP TEST STACK = 700
4726 017566 012705 017604          MOV    #125252,#STBOT 2      ;SET UP NEW R5 VALUE ON STACK
4727 017572 012746 006437          MOV    #1,R5                 ;PUT NEW PC IN R5
4728 017576 000277          MOV    #MARK+37,-(SP)       ;INSERT MARK 37 INSTRUCTION ONTO STACK
4729 017600 000116          SCC
4730                JMP    (SP)                ;* TEST INSTRUCTION
4731 017602 104001          ERROR  +1                    ;MARK INSTRUCTION SHOULD HAVE GONE TO 1#
4732                ;CPU ERROR
4733 017604 101002          ;
4734 017606 100001          1#:  BHI    2#                ;ERROR IF C OR Z BIT CLEAR
4735 017610 102401          BPL    2#                ;ERROR IF N BIT CLEAR
4736                BVS    3#                ;BRANCH IF V BIT SET
4737 017612 104001          2#:  ERROR  +1                    ;BAD CONDITION CODES ON MARK
4738 017614 02270 125252          3#:  CMP    #125252,R5        ;CPU ERROR
4739 017620 001401          BEQ    4#                ;VERIFY R5
4740 017622 104001          ERROR  +1                    ;BRANCH IF GOOD
4741                ;CPU ERROR
4742 017624 020627 001000          4#:  CMP    SP,#STBOT        ;VERIFY THAT STACK IS CORRECT
4743 017630 001401          BEQ    15#               ;BRANCH IF OK
4744                ;ERROR! STACK WAS NOT CORRECT AFTER MARK
4745 017632 104001          ERROR  +1                    ;CPU ERROR
4746                ;CPU ERROR
4747 017634 012746 052525          15#: MOV    #52525,-(SP)       ;SETUP EXPECTED R5
4748 017640 012746 006400          MOV    #6400,-(SP)        ;MOVE MARK 0 INSTRUCTION ON STACK
4749 017644 010605          MOV    SP,R5              ;R5-ADDRESS OF INSTRUCTION
4750 017646 004767 000004          JSR    PC,5#              ;LEAVE 6# ON STACK
4751 017652 000167 000006          6#:  JMP    16#              ;MARK RETURNED CORRECTLY
4752
4753 017656 000257          5#:  CCC
4754 017660 000205          RTS    R5                ;CLEAR THE CONDITION CODES
4755                ;RETURN TO MARK INSTRUCTION
4756                ;NEXT INSTRUCTION ON STACK IS THE RETURN
4757                ;FROM THE JSR
4758 017662 104001          ERROR  +1                    ;ERROR! BAD MARK SEQUENCE
4759                ;CPU ERROR
4760 017664 101402          16#: BLOS   7#                ;IS C OR Z BIT SET?
4761 017666 100401          BMI    7#                ;IS N BIT SET?
4762 017670 102001          BVC    8#                ;IS V BIT SET?
4763                ;ERROR! CONDITIONS CODES INCORRECT
4764 017672 104001          7#:  ERROR  -
4765                ;CPU ERROR
4766 017674 020627 001000          8#:  CMP    R6,#STBOT        ;IS THE STACK CORRECT?
4767 017700 001401          BEQ    9#                ;BRANCH IF YES
4768                ;ERROR! BAD STACK CLEANUP
4769 017702 104001          ERROR  +1                    ;CPU ERROR
4770 017704 022705 052525          9#:  CMP    #52525,R5        ;VERIFY THAT R5 WAS LOADED PROPERLY.
4771 017710 001401          BEQ    10#               ;IF OK, GO TO NEXT TEST.
4772                ;ERROR! R5 WAS NOT CORRECT AFTER MARK.
4773 017712 104001          ERROR  +1                    ;CPU ERROR
4774 017714
4776 017714          10#: XME100:
4778
4779                ; TEST CLEAR CONDITION CODES INSTRUCTION
    
```

***** DOUBLE OPERAND TESTS *****

```

4780 017714 012737 030017 177776      MOV      #30017,0#177776      ;SETUP PSW
4781 017722 000257                      CCC                          ; TEST INSTRUCTION
4782 017724 022737 030000 177776      CMP      #30000,0#177776     ;DID IT CLEAR ALL CONDITION CODE BITS
4783 017732 001401                      BEQ      1#                   ;YES GO ON
4784 017734 104001                      ERROR    +1                   ;CPU ERROR
4785                                     ;NO GO TO ERROR
4786 017736                               1#:
4787                                     ;
4789 017736                               XME. 1:
4791                                     ;
4792                                     ; TEST CLEAR C BIT INSTRUCTION
4793 017736 012737 030017 177776      MOV      #30017,0#177776     ;SETUP PSW
4794 017744 000241                      CLC                          ; TEST INSTRUCTION
4795 017746 022737 030016 177776      CMP      #30016,0#177776     ;DID IT CLEAR CARRY BIT
4796 017754 001401                      BEQ      1#                   ;YES GO ON
4797                                     ;C BIT NOT CLEAR GO TO ERROR
4798 017756 104001                      ERROR    +1                   ;CPU ERROR
4799 017760                               1#:
4800                                     ;
4803 017760                               TE102:
4804                                     ; TEST CLEAR N BIT INST (CLN)
4805 017760 012737 030017 177776      MOV      #30017,0#177776     ;SETUP PSW
4806 017766 000250                      CLN                          ; TEST INSTRUCTION
4807 017770 022737 030007 177776      CMP      #30007,0#177776     ;DID IT CLEAR NEGATIVE BIT
4808 017776 001401                      BEQ      1#                   ;YES GO ON
4809 020000 104001                      ERROR    +1                   ;CPU ERROR
4810                                     ;NO GO TO ERROR
4811 020002                               1#:
4812                                     ;
4815 020002                               TE103:
4816                                     ; TEST CLEAR V BIT INST (CLV)
4817 020002 012737 030017 177776      MOV      #30017,0#177776     ;SETUP PSW
4818 020010 000242                      CLV                          ; TEST INSTRUCTION
4819 020012 022737 030015 177776      CMP      #30015,0#177776     ;DID IT CLEAR OVERFLOW BIT
4820 020020 001401                      BEQ      1#                   ;YES GO ON
4821 020022 104001                      ERROR    +1                   ;CPU ERROR
4822                                     ;NO GO TO ERROR
4823 020024                               1#:
4824                                     ;
4827 020024                               TE104:
4828                                     ; TEST CLEAR Z BIT INST (CLZ)
4829 020024 012737 030017 177776      MOV      #30017,0#177776     ;SETUP PSW
4830 020032 000244                      CLZ                          ; TEST INSTRUCTION
4831 020034 022737 030013 177776      CMP      #30013,0#177776     ;DID IT CLEAR ZERO BIT
4832 020042 001401                      BEQ      1#                   ;YES GO ON
4833 020044 104001                      ERROR    +1                   ;CPU ERROR
4834                                     ;NO GO TO ERROR
4835 020046                               1#:
4836                                     ;
4839 020046                               TE105:
4840                                     ; TEST SET CONDITION CODES INST (SCC)
4841 020046 012737 030000 177776      MOV      #30000,0#177776     ;SETUP PSW
4842 020054 000277                      SCC                          ; TEST INSTRUCTION
4843 020056 022737 030017 177776      CMP      #30017,0#177776     ;DID IT SET ALL CONDITION CODE BITS
4844 020064 001401                      BEQ      1#                   ;YES GO ON
4845 020066 104001                      ERROR    +1                   ;CPU ERROR
4846                                     ;NO GO TO ERROR

```

***** DOUBLE OPERAND TESTS *****

```

4847 020070      1$:
4848             ;
4851 020070      TE106:
4852             ;
4853 020070 012737 030000 177776      ; TEST SET C BIT INST (SEC)
4854 020076 000261                     MOV      #30000,#177776      ;SETUP PSW
4855 020100 022737 030001 177776      SEC                     ; TEST INSTRUCTION
4856 020106 001401                     CMP      #30001,#177776      ;DID IT SET THE CARRY BIT
4857 020110 104001                     BEQ      1$                     ;YES GO ON
4858             ERROR      +1                     ;CPU ERROR
4859 020112             ;NO GO TO ERROR
4860             ;
4863 020112      TE107:
4864             ;
4865 020112 012737 030000 177776      ; TEST SET N BIT INST (SEN)
4866 020120 000270                     MOV      #30000,#177776      ;SETUP PSW
4867 020122 022737 030010 177776      SEN                     ; TEST INSTRUCTION
4868 020130 001401                     CMP      #30010,#177776      ;DID IT SET THE NEGATIVE BIT
4869 020132 104001                     BEQ      1$                     ;YES GO ON
4870             ERROR      +1                     ;CPU ERROR
4871 020134             ;NO GO TO ERROR
4872             ;
4875 020134      TE110:
4876             ;
4877 020134 012737 030000 177776      ; TEST SET V BIT INST (SEV)
4878 020142 000262                     MOV      #30000,#177776      ;SETUP PSW
4879 020144 022737 030002 177776      SEV                     ; TEST INSTRUCTION
4880 020152 001401                     CMP      #30002,#177776      ;DID IT SET THE OVERFLOW BIT
4881 020154 104001                     BEQ      1$                     ;YES GO ON
4882             ERROR      +1                     ;CPU ERROR
4883 020156             ;NO GO TO ERROR
4884             ;
4887 020156      TE111:
4888             ;
4889 020156 012737 030000 177776      ; TEST SET Z BIT INST (SEZ)
4890 020164 000264                     MOV      #30000,#177776      ;SETUP PSW
4891 020166 022737 030004 177776      SEZ                     ; TEST INSTRUCTION
4892 020174 001401                     CMP      #30004,#177776      ;DID IT SET THE ZERO BIT
4893 020176 104001                     BEQ      1$                     ;YES GO ON
4894             ERROR      +1                     ;CPU ERROR
4895 020200             ;NO GO TO ERROR
4896             ;
4899 020200      TE112:
4900             ;
4901 020200 012737 030000 177776      ; TEST MULTIPLE CLEARS OF CC BITS
4902 020206 000277                     MOV      #30000,#177776      ;INIT PSW
4903 020210 000243                     SCC                     ;SETUP PSW
4904 020212 022737 030014 177776      .WORD 243                     ; TEST CLC CLV
4905 020220 001401                     CMP      #30014,#177776      ;PSW CORRECT?
4906 020222 104001                     BEQ      1$                     ;YES GO ON
4907             ERROR      +1                     ;CPU ERROR
4908 020224 000277                     ;NO GO TO ERROR
4909 020226 000245                     SCC                     ;SETUP PSW
4910 020230 022737 030012 177776      .WORD 245                     ; TEST CLC CLZ
4911 020236 001401                     CMP      #30012,#177776      ;PSW CORRECT?
4912 020240 104001                     BEQ      2$                     ;YES GO ON
4913             ERROR      +1                     ;CPU ERROR
4914             ;NO GO TO ERROR

```

***** DOUBLE OPERAND TESTS *****

```

4914 020242 000277      2#:   SCC                      ;SETUP PSW
4915 020244 000246      .WORD 246                ; TEST CLV CLZ
4916 020246 022737 030011 177776  CMP #30011,#177776      ;PSW CORRECT?
4917 020254 001401      BEQ 3#                   ;YES GO ON
4918 020256 104001      ERROR +1                ;CPU ERROR
4919                                     ;NO GO TO ERROR
4920 020260 000277      3#:   SCC                      ;SETUP PSW
4921 020262 000247      .WORD 247                ; TEST CLC CLV CLZ
4922 020264 022737 030010 177776  CMP #30010,#177776      ;PSW CORRECT?
4923 020272 001401      BEQ 4#                   ;YES GO ON
4924 020274 104001      ERROR +1                ;CPU ERROR
4925                                     ;NO GO TO ERROR
4926 020276 000277      4#:   SCC                      ;SETUP PSW
4927 020300 000251      .WORD 251                ; TEST CLN CLC
4928 020302 022737 030006 177776  CMP #30006,#177776      ;PSW CORRECT?
4929 020310 001401      BEQ 5#                   ;YES GO ON
4930 020312 104001      ERROR +1                ;CPU ERROR
4931                                     ;NO GO TO ERROR
4932 020314 000277      5#:   SCC                      ;SETUP PSW
4933 020316 000252      .WORD 252                ; TEST CLN CLV
4934 020320 022737 030005 177776  CMP #30005,#177776      ;PSW CORRECT?
4935 020326 001401      BEQ 6#                   ;YES GO ON
4936 020330 104001      ERROR +1                ;CPU ERROR
4937                                     ;NO GO TO ERROR
4938 020332 000277      6#:   SCC                      ;SETUP PSW
4939 020334 000253      .WORD 253                ; TEST CLN CLC CLV
4940 020336 022737 030004 177776  CMP #30004,#177776      ;PSW CORRECT?
4941 020344 001401      BEQ 7#                   ;YES GO ON
4942 020346 104001      ERROR +1                ;CPU ERROR
4943                                     ;NO GO TO ERROR
4944 020350 000277      7#:   SCC                      ;SETUP PSW
4945 020352 000254      .WORD 254                ; TEST CLN CLZ
4946 020354 022737 030003 177776  CMP #30003,#177776      ;PSW CORRECT?
4947 020362 001401      BEQ 8#                   ;YES GO ON
4948 020364 104001      ERROR +1                ;CPU ERROR
4949                                     ;NO GO TO ERROR
4950 020366 000277      8#:   SCC                      ;SETUP PSW
4951 020370 000255      .WORD 255                ; TEST CLN CLC CLZ
4952 020372 022737 030002 177776  CMP #30002,#177776      ;PSW CORRECT?
4953 020400 001401      BEQ 9#                   ;YES GO ON
4954 020402 104001      ERROR +1                ;CPU ERROR
4955                                     ;NO GO TO ERROR
4956 020404 000277      9#:   SCC                      ;SETUP PSW
4957 020406 000256      .WORD 256                ; TEST CLN CLV CLZ
4958 020410 022737 030001 177776  CMP #30001,#177776      ;SETUP PSW
4959 020416 001401      BEQ 10#                  ;YES GO ON
4960 020420 104001      ERROR +1                ;CPU ERROR
4961                                     ;NO GO TO ERROR
4962 020422                                     10#:
4963                                     ;
4966 020422                                     TE113:
4967                                     ;
4968 020422 012737 030000 177776  MOV #30000,#177776      ;INIT PSW
4969 020430 000263      .WORD 263                ; TEST SEC SEV
4970 020432 022737 030003 177776  CMP #30003,#177776      ;PSW CORRECT?
4971 020440 001401      BEQ 1#                   ;YES GO ON
4972 020442 104001      ERROR +1                ;CPU ERROR

```


***** DOUBLE OPERAND TESTS *****

5032	020642		TE113A:		
5033			:	TEST SIGNED AND CONDITIONAL BRANCHES	
5034	020642	000257		CCC	;CLEAR ALL CC BITS IN PSW
5035	020644	002001		BGE	1\$;BGE SHOULD BRANCH
5036	020646	104001		ERROR	+1 ;CPU ERROR
5037					;ERROR; DIDN'T BRANCH
5038	020650	003001	1\$:	BGT	2\$;BGT SHOULD BRANCH
5039	020652	104001		ERROR	+1 ;CPU ERROR
5040					;ERROR; DIDN'T BRANCH
5041	020654	003401	2\$:	BLE	3\$;BLE SHOULDN'T BRANCH
5042	020656	000401		BR	4\$;BRANCH TO NEXT TEST
5043	020660	104001	3\$:	ERROR	+1 ;CPU ERROR
5044					;ERROR; BLE SHOULD NOT HAVE BRANCHED
5045	020662	002401	4\$:	BLT	5\$;BLT SHOULD NOT BRANCH
5046	020664	000401		BR	6\$;BRANCH TO NEXT TEST
5047	020666	104001	5\$:	ERROR	+1 ;CPU ERROR
5048					;ERROR; BLT SHOULD NOT HAVE BRANCHED
5049	020670	000264	6\$:	SEZ	;SET THE Z BIT IN PSW
5050	020672	003401		BLE	7\$;BLE SHOULD BRANCH
5051	020674	104001		ERROR	+1 ;CPU ERROR
5052					;ERROR; BLE DIDN'T BRANCH
5053	020676	003001	7\$:	BGT	8\$;BGT SHOULD NOT BRANCH
5054	020700	000401		BR	9\$;BRANCH TO NEXT TEST
5055	020702	104001	8\$:	ERROR	+1 ;CPU ERROR
5056					;ERROR; BGT SHOULD NOT HAVE BRANCHED
5057	020704	000257	9\$:	CCC	;CLEAR ALL CC BITS IN PSW
5058	020706	000270		SEN	;SET N BIT IN PSW
5059	020710	002401		BLT	10\$;SHOULD BRANCH TO NEXT TEST
5060	020712	104001		ERROR	+1 ;CPU ERROR
5061					;ERROR; BLT SHOULD HAVE BRANCHED
5062	020714	003401	10\$:	BLE	11\$;SHOULD BRANCH TO NEXT TEST
5063	020716	104001		ERROR	+1 ;CPU ERROR
5064					;ERROR; BLE SHOULD HAVE BRANCHED
5065	020720	002001	11\$:	BGE	12\$;BGE SHOULD NOT BRANCH
5066	020722	000401		BR	13\$;BRANCH TO NEXT TEST
5067	020724	104001	12\$:	ERROR	+1 ;CPU ERROR
5068					;ERROR; BGE SHOULD NOT HAVE BRANCHED
5069	020726	003001	13\$:	BGT	14\$;BGT SHOULD NOT BRANCH
5070	020730	000401		BR	15\$;BRANCH TO NEXT TEST
5071	020732	104001	14\$:	ERROR	+1 ;CPU ERROR
5072					;ERROR; BGT SHOULD NOT HAVE BRANCHED
5073	020734	000257	15\$:	CCC	;CLEAR ALL CC BITS
5074	020736	000262		SEV	;SET V BIT IN PSW
5075	020740	003401		BLE	16\$;BLE SHOULD BRANCH
5076	020742	104001		ERROR	+1 ;CPU ERROR
5077					;ERROR; BLE DIDN'T BRANCH
5078	020744	002401	16\$:	BLT	17\$;BLT SHOULD BRANCH
5079	020746	104001		ERROR	+1 ;CPU ERROR
5080					;ERROR; BLT DIDN'T BRANCH
5081	020750	002001	17\$:	BGE	18\$;BGE SHOULDN'T BRANCH
5082	020752	000401		BR	19\$;BRANCH TO NEXT TEST
5083	020754	104001	18\$:	ERROR	+1 ;CPU ERROR
5084					;ERROR; BGE SHOULD NOT HAVE BRANCHED
5085	020756	003001	19\$:	BGT	20\$;BGT SHOULD NOT BRANCH
5086	020760	000401		BR	21\$;BRANCH TO NEXT TEST
5087	020762	104001	20\$:	ERROR	+1 ;CPU ERROR
5088					;ERROR; BGT SHOULD NOT HAVE BRANCHED

***** DOUBLE OPERAND TESTS *****

```

5089 020764 000257      21$:   CCC                ;CLEAR ALL CC BITS
5090 020766 000272      .WORD 272             ;SET N AND V BITS IN PSW
5091 020770 002001      BGE 22$              ;BGE SHOULD BRANCH
5092 020772 104001      ERROR +1            ;CPU ERROR
5093                                     ;ERROR; BGE DIDN'T BRANCH
5094 020774 003001      22$:   BGT 23$              ;BGT SHOULD BRANCH
5095 020776 104001      ERROR +1            ;CPU ERROR
5096                                     ;ERROR; BGT DIDN'T BRANCH
5097 021000 003401      23$:   BLE 24$              ;BLE SHOULDN'T BRANCH
5098 021002 000401      BR 25$              ;BRANCH TO NEXT TEST
5099 021004 104001      24$:   ERROR +1            ;CPU ERROR
5100                                     ;ERROR; BLE SHOULD NOT HAVE BRANCHED
5101 021006 002401      25$:   BLT 26$              ;BLT SHOULD NOT BRANCH
5102 021010 000401      BR 27$              ;BRANCH TO NEXT TEST
5103 021012 104001      26$:   ERROR +1            ;CPU ERROR
5104                                     ;ERROR; BLT SHOULD NOT HAVE BRANCHED
5105 021014      27$:
5108 021014      TE114:
5109      ;
5110 021014 012737 030000 177776      ; TEST NOP INST
5111 021022 012700 000001      MOV #30000,#177776      ;INIT PSW
5112 021026 012701 000002      MOV #1,R0              ;INIT R0
5113 021032 012702 000003      MOV #2,R1              ;INIT R1
5114 021036 012703 000004      MOV #3,R2              ;INIT R2
5115 021042 012704 000005      MOV #4,R3              ;INIT R3
5116 021046 012705 000006      MOV #5,R4              ;INIT R4
5117 021052 010637 003012      MOV #6,R5              ;INIT R5
5118 021056 012737 030017 114146      MOV R6,#SLOC00        ;SAVE SP
5119 021064 000277      SCC                   ;SETUP PSW
5120 021066 000240      NOP                   ;SET ALL CONDITION CODE BITS
5121 021070 000240      NOP                   ; TEST INSTRUCTION
5122 021072 000240      NOP                   ; TEST INSTRUCTION
5123 021074 004767 000046      JSR PC,T114           ; TEST INSTRUCTION
5124 021100 020637 003012      CMP R6,#SLOC00        ;CHECK PSW, AND GPR'S
5125 021104 001401      BEQ 1$                ;CHECK SP
5126 021106 104001      ERROR +1            ;OK GO ON
5127                                     ;CPU ERROR
5128 021110 012737 030000 114146 1$:  MOV #30000,#EXPDAT    ;NO GO TO ERROR
5129 021116 000257      CCC                   ;SETUP PSW
5130 021120 000260      .WORD 260            ;CLEAR ALL CONDITION CODE BITS
5131 021122 000260      .WORD 260            ; TEST FOR NOP OPERATION
5132 021124 000260      .WORD 260            ; TEST FOR NOP OPERATION
5133 021126 004767 000014      JSR PC,T114           ; TEST FOR NOP OPERATION
5134 021132 020637 003012      CMP R6,#SLOC00        ;CHECK PSW, AND GPR'S
5135 021136 001401      BEQ 2$                ;CHECK SP
5136 021140 104001      ERROR +1            ;OK GO ON
5137                                     ;CPU ERROR
5138 021142      2$:
5139 021142 000167 000104      JMP FINNOP            ;NO GO TO ERROR
5140      ;
5141 021146 023737 114146 177776      T114:  CMP #EXPDAT,#177776  ;CHECK PSW
5142 021154 001405      BEQ TA114            ;OK GO ON
5143 021156 010067 157216      MOV R0,400           ;SAVE R0
5144 021162 104001      ERROR +1            ;CPU ERROR
5145                                     ;NO GO TO ERROR
5146 021164 016700 157210      MOV 400,R0           ;RESTORE R0
5147 021170 022700 000001      TA114:  CMP #1,R0        ;CHECK R0

```

***** DOUBLE OPERAND TESTS *****

```

5148 021174 001401          BEQ    TB114          ;OK GO ON
5149 021176 104001          ERROR   +1          ;CPU ERROR
5150                                     ;NO GO TO ERROR
5151 021200 022701 000002    TB114:  CMP    #2,R1      ;CHECK R1
5152 021204 001401          BEQ    TC114          ;OK GO ON
5153 021206 104001          ERROR   +1          ;CPU ERROR
5154                                     ;NO GO TO ERROR
5155 021210 022702 000003    TC114:  CMP    #3,R2      ;CHECK R2
5156 021214 001401          BEQ    TD114          ;OK GO ON
5157 021216 104001          ERROR   +1          ;CPU ERROR
5158                                     ;NO GO TO ERROR
5159 021220 022703 000004    TD114:  CMP    #4,R3      ;CHECK R3
5160 021224 001401          BEQ    TF114          ;OK GO ON
5161 021226 104001          ERROR   +1          ;CPU ERROR
5162                                     ;NO GO TO ERROR
5163 021230 022704 000005    TF114:  CMP    #5,R4      ;CHECK R4
5164 021234 001401          BEQ    TG114          ;OK GO ON
5165 021236 104001          ERROR   +1          ;CPU ERROR
5166                                     ;NO GO TO ERROR
5167 021240 022705 000006    TG114:  CMP    #6,R5      ;CHECK R5
5168 021244 001401          BEQ    TH114          ;OK GO ON
5169 021246 104001          ERROR   +1          ;CPU ERROR
5170                                     ;NO GO TO ERROR
5171 021250 000207          TH114:  RTS    PC          ;RETURN
5172                                     ;
5173 021252 000240          ;FINNOP· NOP
5174                                     ;
5177                                     ;
5178                                     ;-----
5179                                     ;TEST ALTERNATE REGISTER SET
5180                                     ;
5181 021254 005000          CLR    R0          ;-----CLEAR-----
5182 021256 005001          CLR    R1          ;-----PRIMARY-----
5183 021260 005002          CLR    R2          ;-----GENERAL-----
5184 021262 005003          CLR    R3          ;-----PURPOSE-----
5185 021264 005004          CLR    R4          ;-----REGISTER-----
5186 021266 005005          CLR    R5          ;-----SET-----
5187 021270 012767 004000 156500  MOV    #4000,PS     ;SELECT ALTERNATE REGISTER SET
5188 021276 032767 004000 156472  BIT    #BIT11,PS    ;TEST TO SEE THAT BIT IS SET IN PS
5189 021304 001001          BNE    1$          ;IF IT'S SET GO TESTS REGISTERS
5190 021306 104001          ERROR   +1          ;CPU ERROR
5191                                     ;ERROR! ALTERNATE REGISTER SELECT
5192                                     ;BIT IN PS IS NOT SET
5193 021310 012700 177777    1$:   MOV    #177777,R0  ;WRITE ALL ONES TO ALTERNATE REG SET
5194 021314 010001          MOV    R0,R1
5195 021316 010102          MOV    R1,R2
5196 021320 010203          MOV    R2,R3
5197 021322 010304          MOV    R3,R4
5198 021324 010405          MOV    R4,R5
5199 021326 042767 004000 156442  BIC    #BIT11,PS    ;CLEAR BIT 11 IN PS
5200 021334 032767 004000 156434  BIT    #BIT11,PS    ;IS BIT 11 = 0?
5201 021342 001401          BEQ    2$          ;IF IT'S NOT ZERO
5202 021344 104001          ERROR   +1          ;CPU ERROR
5203                                     ;ERROR! BIT 11 DID NOT CLEAR
5204 021346 005700          2$:   TST    R0          ;MAKE SURE THAT WE REALLY
5205 021350 001401          BEQ    3$          ;WERE TALKING TO ALTERNATE REGISTERS.
5206 021352 104001          ERROR   +1          ;CPU ERROR

```

***** DOUBLE OPERAND TESTS *****

```

5207
5208 021354 005701          3#:  TST    R1          ;ERROR!
5209 021356 001401          BEQ    4#          ;
5210 021360 104001          ERROR  +1          ;CPU ERROR
5211                                     ;ERROR!
5212 021362 005702          4#:  TST    R2          ;
5213 021364 001401          BEQ    5#          ;
5214 021366 104001          ERROR  +1          ;CPU ERROR
5215                                     ;ERROR!
5216 021370 005703          5#:  TST    R3          ;
5217 021372 001401          BEQ    6#          ;
5218 021374 104001          ERROR  +1          ;CPU ERROR
5219                                     ;
5220 021376 005704          6#:  TST    R4          ;
5221 021400 001401          BEQ    7#          ;
5222 021402 104001          ERROR  +1          ;CPU ERROR
5223                                     ;
5224 021404 005705          7#:  TST    R5          ;
5225 021406 001401          BEQ    8#          ;
5226 021410 104001          ERROR  +1          ;CPU ERROR
5227                                     ;ERROR!
5228 021412 012767 004000 156356 8#:  MOV    #BIT11,PS  ;ENABLE ALTERNATE REGISTER SET
5229 021420 005000          CLR    R0          ;-- --CLEAR-- --
5230 021422 005001          CLR    R1          ;---ALTERNATE---
5231 021424 005002          CLR    R2          ;---REGISTER---
5232 021426 005003          CLR    R3          ;-----SET-----
5233 021430 005004          CLR    R4          ;
5234 021432 005005          CLR    R5          ;
5235 021434 C42767 004000 156334 BIC    #BIT11,PS  ;BACK TO PRIMARY GPRS
5236 021442 012700 177777 MOV    #177777,R0 ;ENSURE THAT WRITES TO PRIMARY GPRS
5237 021446 010001          MOV    R0,R1      ;DO NOT EFFECT ALTERNATE GPRS
5238 021450 010102          MOV    R1,R2      ;
5239 021452 010203          MOV    R2,R3      ;
5240 021454 010304          MOV    R3,R4      ;
5241 021456 010405          MOV    R4,R5      ;
5242 021460 052767 004000 156310 BIS    #BIT11,PS  ;RETURN TO ALTERNATE REGISTERS
5243 021466 005700          TST    R0          ;SHOULD BE ALL 0'S
5244 021470 001401          BEQ    9#          ;
5245 021472 104001          ERROR  +1          ;CPU ERROR
5246                                     ;ERROR!
5247 021474 005701          9#:  TST    R1          ;
5248 021476 001401          BEQ   10#         ;
5249 021500 104001          ERROR  +1          ;CPU ERROR
5250                                     ;
5251 021502 005702          10#: TST    R2          ;
5252 021504 001401          BEQ   11#         ;
5253 021506 104001          ERROR  +1          ;CPU ERROR
5254                                     ;
5255 021510 005703          11#: TST    R3          ;
5256 021512 001401          BEQ   12#         ;
5257 021514 104001          ERROR  +1          ;CPU ERROR
5258                                     ;
5259 021516 005704          12#: TST    R4          ;
5260 021520 001401          BEQ   13#         ;
5261 021522 104001          ERROR  +1          ;CPU ERROR
5262                                     ;
5263 021524 005705          13#: TST    R5          ;

```

***** DOUBLE OPERAND TESTS *****

```

5264 021526 001401          BEQ      14$
5265 021530 104001          ERROR    +1          ;CPU ERROR
5266 021532                14$:
5267
5268 021532          ALR0TS:
5269          ; ALTERNATE REGISTER SET R0 BIT TESTS
5270 021532 012700 177777    MOV      #177777,R0          ;R0=177777
5271 021536 020027 177777    CMP      R0,#177777        ;DOES R0=177777
5272 021542 001401          BEQ      1$              ;YES GO ON
5273 021544 104001          ERROR    +1          ;CPU ERROR
5274
5275 021546 005000          1$:      CLR      R0              ;NO GO TO ERROR
5276 021550 020027 000000    CMP      R0,#0            ;R0=0
5277 021554 001401          BEQ      2$              ;DOES R0=0
5278 021556 104001          ERROR    +1          ;YES GO ON
5279
5280 021560 012700 125252    2$:      MOV      #125252,R0        ;NO GO TO ERROR
5281 021564 020027 125252    CMP      R0,#125252       ;R0=125252
5282 021570 001401          BEQ      3$              ;DOES R0=125252
5283 021572 104001          ERROR    +1          ;YES GO ON
5284
5285 021574 012700 052525    3$:      MOV      #52525,R0        ;NO GO TO ERROR
5286 021600 020027 052525    CMP      R0,#52525        ;R0=52525
5287 021604 001401          BEQ      4$              ;DOES R0=52525
5288 021606 104001          ERROR    +1          ;YES GO ON
5289
5290 021610          4$:
5291          ;
5294 021610          ALR1TS:
5295          ; ALTERNATE REGISTER SET R1 BIT TESTS
5296 021610 012701 177777    MOV      #177777,R1        ;R1=177777
5297 021614 020127 177777    CMP      R1,#177777       ;DOES R1=177777
5298 021620 001401          BEQ      1$              ;YES GO ON
5299 021622 104001          ERROR    +1          ;CPU ERROR
5300
5301 021624 005001          1$:      CLR      R1              ;NO GO TO ERROR
5302 021626 020127 000000    CMP      R1,#0            ;R1=0
5303 021632 001401          BEQ      2$              ;DOES R1=0
5304 021634 104001          ERROR    +1          ;YES GO ON
5305
5306 021636 012701 125252    2$:      MOV      #125252,R1        ;NO GO TO ERROR
5307 021642 020127 125252    CMP      R1,#125252       ;R1=125252
5308 021646 001401          BEQ      3$              ;DOES R1=125252
5309 021650 104001          ERROR    +1          ;YES GO ON
5310
5311 021652 012701 052525    3$:      MOV      #52525,R1        ;NO GO TO ERROR
5312 021656 020127 052525    CMP      R1,#52525        ;R1=52525
5313 021662 001401          BEQ      4$              ;DOES R1=52525
5314 021664 104001          ERROR    +1          ;YES GO ON
5315
5316 021666          4$:
5317          ;
5320 021666          ALR2TS:
5321          ; ALTERNATE REGISTER SET R2 BIT TESTS
5322 021666 012702 177777    MOV      #177777,R2        ;R2=177777
5323 021672 020227 177777    CMP      R2,#177777       ;DOES R2=177777
5324 021676 001401          BEQ      1$              ;YES GO ON

```

***** DOUBLE OPERAND TESTS *****

```

5325 021700 104001          ERROR +1          ;CPU ERROR
5326                                     ;NO GO TO ERROR
5327 021702 005002      1$: CLR R2          ;R2=0
5328 021704 020227 000000  CMP R2,#0      ;DOES R2=0
5329 021710 001401      BEQ 2$          ;YES GO ON
5330 021712 104001          ERROR +1          ;CPU ERROR
5331                                     ;NO GO TO ERROR
5332 021714 012702 125252  2$: MOV #125252,R2 ;R2=125252
5333 021720 020227 125252  CMP R2,#125252 ;DOES R2=125252
5334 021724 001401      BEQ 3$          ;YES GO ON
5335 021726 104001          ERROR +1          ;CPU ERROR
5336                                     ;NO GO TO ERROR
5337 021730 012702 052525  3$: MOV #52525,R2 ;R2=52525
5338 021734 020227 052525  CMP R2,#52525 ;DOES R2=52525
5339 021740 001401      BEQ 4$          ;YES GO ON
5340 021742 104001          ERROR +1          ;CPU ERROR
5341                                     ;NO GO TO ERROR
5342 021744      4$:
5343 ;
5346 021744      ;ALR3TS:
5347 ;
5348 021744 012703 177777  ;
5349 021750 020327 177777  MOV #177777,R3 ;R3=177777
5350 021754 001401      BEQ 1$          ;DOES R3=177777
5351 021756 104001          ERROR +1          ;YES GO ON
5352                                     ;CPU ERROR
5353 021760 005003      1$: CLR R3          ;NO GO TO ERROR
5354 021762 020327 000000  CMP R3,#0      ;R3=0
5355 021766 001401      BEQ 2$          ;DOES R3=0
5356 021770 104001          ERROR +1          ;YES GO ON
5357                                     ;CPU ERROR
5358 021772 012703 125252  2$: MOV #125252,R3 ;NO GO TO ERROR
5359 021776 020327 125252  CMP R3,#125252 ;R3=125252
5360 022002 001401      BEQ 3$          ;DOES R3=125252
5361 022004 104001          ERROR +1          ;YES GO ON
5362                                     ;CPU ERROR
5363 022006 012703 052525  3$: MOV #52525,R3 ;NO GO TO ERROR
5364 022012 020327 052525  CMP R3,#52525 ;R3=52525
5365 022016 001401      BEQ 4$          ;DOES R3=52525
5366 022020 104001          ERROR +1          ;YES GO ON
5367                                     ;CPU ERROR
5368 022022      4$:
5369 ;
5372 022022      ;ALR4TS:
5373 ;
5374 022022 012704 177777  ;
5375 022026 020427 177777  MOV #177777,R4 ;R4=177777
5376 022032 001401      BEQ 1$          ;DOES R4=177777
5377 022034 104001          ERROR +1          ;YES GO ON
5378                                     ;CPU ERROR
5379 022036 005004      1$: CLR R4          ;NO GO TO ERROR
5380 022040 020427 000000  CMP R4,#0      ;R4=0
5381 022044 001401      BEQ 2$          ;DOES R4=0
5382 022046 104001          ERROR +1          ;YES GO ON
5383                                     ;CPU ERROR
5384 022050 012704 125252  2$: MOV #125252,R4 ;NO GO TO ERROR
5385 022054 020427 125252  CMP R4,#125252 ;R4=125252

```


***** DOUBLE OPERAND TESTS *****

```

5386 022060 001401      BEQ      3$      ;YES GO ON
5387 022062 104001      ERROR    +1      ;CPU ERROR
5388                                     ;NO GO TO ERROR
5389 022064 012704 052525 3$:      MOV      @52525,R4      ;R4=52525
5390 022070 020427 052525      CMP      R4,@52525      ;DOES R4=52525
5391 022074 001401      BEQ      4$      ;YES GO ON
5392 022076 104001      ERROR    +1      ;CPU ERROR
5393                                     ;NO GO TO ERROR
5394 022100      4$:
5395      ;
5398 022100      ALR5TS:
5399      ;
5400 022100 012705 177777      ; ALTERNATE REGISTER SET R5 BIT TESTS
5401 022104 020527 177777      MOV      @177777,R5      ;R5=177777
5402 022110 001401      CMP      R5,@177777      ;DOES R5=177777
5403 022112 104001      BEQ      1$      ;YES GO ON
5404      ERROR    +1      ;CPU ERROR
5405      ;NO GO TO ERROR
5406 022114 005005 000000 1$:      CLR      R5      ;R5=0
5407 022116 020527      CMP      R5,@0      ;DOES R5=0
5408 022124 104001      BEQ      2$      ;YES GO ON
5409      ERROR    +1      ;CPU ERROR
5410 022126 012705 125252 2$:      MOV      @125252,R5      ;R5=125252
5411 022132 020527 125252      CMP      R5,@125252      ;DOES R5=125252
5412 022136 001401      BEQ      3$      ;YES GO ON
5413 022140 104001      ERROR    +1      ;CPU ERROR
5414      ;NO GO TO ERROR
5415 022142 012705 052525 3$:      MOV      @52525,R5      ;R5=52525
5416 022146 020527 052525      CMP      R5,@52525      ;DOES R5=52525
5417 022152 001401      BEQ      4$      ;YES GO ON
5418 022154 104001      ERROR    +1      ;CPU ERROR
5419      ;NO GO TO ERROR
5420 022156 042767 004000 155612 4$:      BIC      @BIT11,PS      ;RETURN TO PRIMARY GEN PURPOSE REGS
5421      ;
5422      ;
5425 022164      TE115:
5426      ;
5427 022164 005004      ; TEST MOVE FROM PROCCESOR STATUS INST (MFPS)
5428      CLR      R4      ;SETUP DESTINATION R4
5429 022166 012701 022374      MOV      @TE115A,R1      ;SETUP POINTERS TO TABLES
5430 022172 012702 022404      MOV      @TE115B,R2      ;
5431 022176 012703 022412      MOV      @TE115C,R3      ;
5432 022202 012137 177776 1$:      MOV      (R1)+,@177776      ;SETUP PSW
5433 022206 106704      MFPS     R4      ; TEST INSTRUCTION
5434 022210 023722 177776      CMP      @@177776,(R2)+      ;CHECK PSW
5435 022214 001401      BEQ      2$      ;OK GO ON
5436 022216 104001      ERROR    +1      ;CPU ERROR
5437      ;NO GO TO ERROR
5438 022220 020423 2$:      CMP      R4,(R3)+      ;CHECK R4
5439 022222 001401      BEQ      3$      ;OK GO ON
5440 022224 104001      ERROR    +1      ;CPU ERROR
5441      ;NO GO TO ERROR
5442 022226 021127 177777 3$:      CMP      (R1),@177777      ;ARE WE DONE
5443 022232 001363      BNE     1$      ;NO GO TO 1$
5444      ;
5445      ;
5446 022234 012701 022374      MOV      @TE115A,R1      ;SETUP POINTERS TO TABLES

```

***** DOUBLE OPERAND TESTS *****

```

5447 022240 012702 022404      MOV    #TE115B,R2      ;
5448 022244 012703 022412      MOV    #TE115C,R3      ;
5449 022250 010605              MOV    R6,R5           ;SAVE STACK IN R5
5450 022252 011137 177776      101$: MOV    (R1),#177776 ;SETUP PSW
5451 022256 106706              MFPS   R6              ; TEST INSTRUCTION
5452 022260 023712 177776      CMP    #177776,(R2)   ;CHECK PSW
5453 022264 001401              BEQ    102$           ;OK GO ON
5454 022266 104001              ERROR  +1            ;CPU ERROR
5455                          ;NO GO TO ERROR
5456 022270 020613              102$: CMP    R6,(R3)   ;CHECK R6
5457 022272 001401              BEQ    103$           ;OK GO ON
5458 022274 104001              ERROR  +1            ;CPU ERROR
5459                          ;NO GO TO ERROR
5460 022276 010506              103$: MOV    R5,R6     ;RESTORE STACK
5461
5462
5463 022300 012701 022374      MOV    #TE115A,R1      ;SETUP POINTERS TO TABLES
5464 022304 012702 022404      MOV    #TE115B,R2      ;
5465 022310 012703 022420      MOV    #TE115D,R3      ;
5466 022314 005037 114146      CLR    #EXPDAT         ;INIT EXPECTED DATA HOLDER.
5467 022320 012704 114146      4$:  MOV    #EXPDAT,R4   ;SETUP POINTER TO TEST LOCATION
5468 022324 012137 177776      MOV    (R1)+,#177776  ;SETUP PSW
5469 022330 106724              MFPS   (R4)+          ; TEST INSTRUCTION
5470 022332 023722 177776      CMP    #177776,(R2)+  ;CHECK PSW
5471 022336 001401              BEQ    5$            ;OK GO ON
5472 022340 104001              ERROR  +1            ;CPU ERROR
5473                          ;NO GO TO ERROR
5474 022342 020427 114147      5$:  CMP    R4,#EXPDAT+1 ;CHECK R4
5475 022346 001401              BEQ    6$            ;OK GO ON
5476 022350 104001              ERROR  +1            ;CPU ERROR
5477                          ;NO GO TO ERROR
5478 022352 023723 114146      6$:  CMP    #EXPDAT,(R3)+ ;CHECK TEST LOCATION
5479 022356 001401              BEQ    7$            ;OK GO ON
5480 022360 104001              ERROR  +1            ;CPU ERROR
5481                          ;NO GO TO ERROR
5482 022362 021127 177777      7$:  CMP    (R1),#177777 ;ARE WE DONE
5483 022366 001354              BNE    4$            ;NO GO TO 4$
5484
5485 022370 000167 000032      JMP    TE115F
5486
5487 ;
5488 ;
5489 022374 030207      TE115A: .WORD 30207
5490 022376 030000      .WORD 30000
5491 022400 030057      .WORD 30057
5492 022402 177777      .WORD 177777
5493 022404 030211      TE115B: .WORD 30211
5494 022406 030004      .WORD 30004
5495 022410 030041      .WORD 30041
5496 022412 177607      TE115C: .WORD 177607
5497 022414 000000      .WORD 0
5498 022416 000057      .WORD 57
5499 022420 000207      TE115D: .WORD 207
5500 022422 000000      .WORD 0
5501 022424 000057      .WORD 57
5502 022426 000240      TE115F: NOP
5503 ;

```

***** DOUBLE OPERAND TESTS *****

```

5506 022430          TE116:
5507                ; TEST MOVE TO PROCESSOR STATUS INST (MTPS)
5508
5509 022430 012737 030000 177776      MOV    #30000,#177776      ;SET PSW TO KERNEL MODE
5510 022436 012701 022732              MOV    #TE116D,R1        ;SETUP POINTERS TO TABLES
5511 022442 012702 022710              MOV    #TE116B,R2
5512 022446 010103                    MOV    R1,R3
5513 022450 004767 000136              JSR    PC,T116          ; TEST INSTRUCTION AND CHECK PSW
5514                                          ;AND SOURCE OPERAND
5515
5516
5517 022454 012737 140000 177776      MOV    #140000,#177776   ;SET PSW TO USER MODE
5518 022462 012706 000600              MOV    #600,R6          ;SETUP USER STACK
5519 022466 012701 022732              MOV    #TE116D,R1        ;SETUP POINTERS TO TABLES
5520 022472 012702 022722              MOV    #TE116C,R2
5521 022476 010103                    MOV    R1,R3
5522 022500 004767 000106              JSR    PC,T116          ; TEST INSTRUCTION AND CHECK PSW
5523                                          ;AND SOURCE OPERAND
5524
5525
5526 022504 012737 140000 177776      MOV    #140000,#177776   ;SET PSW TO USER MODE AND CLEAR CC BITS
5527 022512 012701 022704              MOV    #TE116A,R1        ;SETUP POINTERS TO TABLES
5528 022516 012702 022722              MOV    #TE116C,R2
5529 022522 012703 022732              MOV    #TE116D,R3
5530 022526 010104                    MOV    R1,R4          ;SAVE A COPY OF R1 INTO R4
5531 022530 004767 000110              JSR    PC,TA116         ; TEST INSTRUCTION AND CHECK PSW
5532                                          ;AND SOURCE OPERAND
5533
5534
5535 022534 012737 030000 177776      MOV    #30000,#177776   ;SET PSW TO KERNEL MODE
5536 022542 012701 022704              MOV    #TE116A,R1        ;SETUP POINTERS TO TABLES
5537 022546 012702 022710              MOV    #TE116B,R2
5538 022552 012703 022732              MOV    #TE116D,R3
5539 022556 010104                    MOV    R1,R4          ;SAVE A COPY OF R1 INTO R4
5540 022560 004767 000060              JSR    PC,TA116         ; TEST INSTRUCTION AND CHECK PSW
5541                                          ;AND SOURCE OPERAND
5542
5543 022564 005037 177776              CLR    #177776          ;SET PSW TO KERNEL MODE
5544 022570 106427 177412              MTPS   #177412          ;TEST INSTRUCTION
5545 022574 022737 000012 177776      CMP    #12,#177776      ;IS PSW CORRECT
5546 022602 001401                    BEQ    100#             ;YES GO ON
5547 022604 104001                    ERROR  +1              ;CPU ERROR
5548                                          ;NO GO TO ERROR
5549                100#:
5550 022606 000167 000130              JMP    FIN116
5551                ;
5552 022612 012105          T116:  MOV    (R1),R5          ;MOVE TEST DATA TO R5
5553 022614 106405          MTPS   R5              ; TEST INSTRUCTION
5554 022616 023722 177776      CMP    #177776,(R2)    ;IS PSW CORRECT
5555 022622 001401          BEQ    1#              ;YES GO ON
5556 022624 104001          ERROR  +1              ;CPU ERROR
5557                                          ;NO GO TO ERROR
5558 022626 022305          1#:  CMP    (R3),R5          ;IS R5 CORRECT
5559 022630 001401          BEQ    2#              ;YES GO ON
5560 022632 104001          ERROR  +1              ;CPU ERROR
5561                                          ;NO GO TO ERROR
5562 022634 021227 177777          2#:  CMP    (R2),#177777   ;ARE WE DONE

```

***** DOUBLE OPERAND TESTS *****

```

5563 022640 001364      BNE      T116      ;NO GO TO T116
5564 022642 000207      RTS       PC        ;RETURN
5565
5566 022644 106421      ;
5567 022646 023722 177776 ;TA116: MTPS    (R1).      ; TEST INSTRUCTION
5568 022652 001401      CMP      @#177776,(R2)  ; IS PSW CORRECT
5569 022654 104001      BEQ      1#         ; YES GO ON
5570
5571 022656 122423      1#:     CMPB    (R4)+,(R3)+ ;CPU ERROR
5572 022660 001401      BEQ      2#         ;NO GO TO ERROR
5573 022662 104001      ERROR   +1        ;CHECK TEST LOCATION
5574
5575 022664 020104      2#:     CMP      R1,R4    ;OK GO ON
5576 022666 001401      BEQ      3#         ;CPU ERROR
5577 022670 104001      ERROR   +1        ;NO GO TO ERROR
5578
5579 022672 005203      3#:     INC      R3      ;IS SOURCE OPERAND CORRECT
5580 022674 021227 177777 ;      CMP      (R2),@#177777 ; YES GO ON
5581 022700 001361      BNE      TA116     ;CPU ERROR
5582 022702 000207      RTS       PC        ;NO GO TO ERROR
5583
5584 022704      377      ;TE116A: .BYTE 377      ;POINT TO NEXT WORD
5585 022705      000      .BYTE 0          ;ARE WE DONE
5586 022706      252      .BYTE 252       ;NO GO TO TA116
5587 022707      125      .BYTE 125       ;RETURN
5588 022710 030357      TE116B: .WORD 30357
5589 022712 030000      .WORD 30000
5590 022714 030252      .WORD 30252
5591 022716 030105      .WORD 30105
5592 022720 177777      .WORD 177777
5593 022722 140017      TE116C: .WORD 140017
5594 022724 140000      .WORD 140000
5595 022726 140012      .WORD 140012
5596 022730 140005      .WORD 140005
5597 022732 177777      TE116D: .WORD 177777
5598 022734 177400      .WORD 177400
5599 022736 177652      .WORD 177652
5600 022740 177525      .WORD 177525
5601
5602 022742      ;FIN116:
5603
5613 022742      ;TE117:
5614
5615 022742 013746 000010 ;      TEST MFPT (MOVE FROM PROCESSOR TYPE)
5616 022746 012737 023054 000010 ;MOV @#10,-(SP) ;SAVE VECTOR
5617 022754 012700 177777 ;MOV @TE117A,@#10 ;SETUP VECTOR TO HANDLE POSSIBLE ILLEGAL INST TRAP
5618 022760 012737 030000 177776 ;MOV @177777,R0 ;INIT R0
5619 022766 000007 ;MOV @30000,@#177776 ;SETUP PSW
5620 022770 022737 030000 177776 ;.WORD 7 ; TEST INSTRUCTION
5621 022776 001401 ;CMP @30000,@#177776 ; IS PSW CORRECT
5622 023000 104001 ;BEQ 1# ; YES GO ON
5623 ;ERROR +1 ;CPU ERROR
5624 023002 020027 000005 ;1#: CMP R0,#5 ;NO GO TO ERROR
5625 023006 001401 ;BEQ 2# ;IS R0 CORRECT
5626 023010 104001 ;ERROR +1 ;YES GO ON
5627 ;CPU ERROR
5628 023012 012700 177777 ;2#: MOV @177777,R0 ;NO GO TO ERROR
;INIT R0

```


***** DOUBLE OPERAND TESTS *****

```

5699          ;      CMPB   #APTENV,#ENV          ;ARE WE IN APT MODE?
5700          ;      BNE     1#                    ;IF NOT: DO THIS TEST
5701 023210   005767 155772          ;      TST     $PASS          ;FIRST PASS??
5702 023214   001402          ;      BEQ     1#                    ;IF YES, DO IT
5703 023216   000167 000622          ;      JMP     FIN122          ;ELSE SKIP THIS TEST BECAUSE RESETS
5704          ;                               ;SCREW UP THE APT MONITOR.
5705 023222   012737 030340 177776 1#:   MOV     #30340,#177776          ;SETUP PSW TO KERNEL MODE
5706 023230   012737 160000 177572          ;      MOV     #160000,#177572      ;SETUP MMRO
5707 023236   012737 000077 172516          ;      MOV     #77,#172516          ;SETUP MMR3
5708 023244   005037 177772          ;      CLR     #177772          ;CLEAR PIRQ
5709 023250   023727 177772 000000          ;      CMP     #177772,#0          ;IS PIRQ CORRECT
5710 023256   001401          ;      BEQ     C121A          ;YES GO ON
5711 023260   104001          ;      ERROR   +1                ;CPU ERROR
5712          ;                               ;NO GO TO ERROR
5713 023262   012737 025000 177772 C121A: MOV     #25000,#177772          ;MOVE AN ALTERNATING PATTERN TO PIRQ
5714 023270   022737 025252 177772          ;      CMP     #25252,#177772      ;IS PIRQ CORRECT
5715 023276   001401          ;      BEQ     C121B          ;YES GO ON
5716 023300   104001          ;      ERROR   +1                ;CPU ERROR
5717          ;                               ;NO GO TO ERROR
5718 023302   012737 077000 177772 C121B: MOV     #77000,#177772          ;SETUP PIRQ
5719 023310   022737 077314 177772          ;      CMP     #77314,#177772      ;IS PIRQ CORRECT
5720 023316   001401          ;      BEQ     C121C          ;YES GO ON
5721 023320   104001          ;      ERROR   +1                ;CPU ERROR
5722          ;                               ;NO GO TO ERROR
5723 023322   000277          ;      SCC     C121C:          ;SET ALL CC BITS
5724 023324   000005          ;      RESET          ; TEST INSTRUCTION
5725 023326   022737 030357 177776          ;      CMP     #30357,#177776      ;IS PSW CORRECT
5726 023334   001401          ;      BEQ     1#                    ;YES GO ON
5727 023336   104001          ;      ERROR   +1                ;CPU ERROR
5728          ;                               ;NO GO TO ERROR
5729 023340   013701 177572          1#:   MOV     #SRO,R1          ;SAVE SRO IN R1
5730 023344   042701 000176          ;      BIC     #176,R1          ;STRIP OFF UNDEFINED BITS 1 6 FROM MMRO
5731 023350   022701 000000          ;      CMP     #0,R1          ;IS MMRO CORRECT
5732 023354   001401          ;      BEQ     2#                    ;YES GO ON
5733 023356   104001          ;      ERROR   +1                ;CPU ERROR
5734          ;                               ;NO GO TO ERROR
5735 023360   022737 000000 172516 2#:   CMP     #0,#172516          ;IS MMR3 CORRECT
5736 023366   001401          ;      BEQ     3#                    ;YES GO ON
5737 023370   104001          ;      ERROR   +1                ;CPU ERROR
5738          ;                               ;NO GO TO ERROR
5739 023372   022737 000000 177772 3#:   CMP     #0,#177772          ;IS PIRQ CORRECT
5740 023400   001401          ;      BEQ     4#                    ;YES GO ON
5741 023402   104001          ;      ERROR   +1                ;CPU ERROR
5742          ;                               ;NO GO TO ERROR
5743          ;
5744 023404   013702 000004          4#:   MOV     #4,R2          ;SAVE LOC 4 IN R2
5745 023410   013703 000006          ;      MOV     #6,R3          ;SAVE LOC 6 IN R3
5746 023414   012737 023552 000004          ;      MOV     #8,#4          ;SETUP VECTORS IN CASE AN ERROR CAUSES
5747          ;                               ;A HALT TO OCCUR IN USER MODE.
5748 023422   012737 000340 000006          ;      MOV     #340,#6          ;THIS SETS KERNEL MODE, AND PREVENTS PIRQ
5749          ;                               ;INTERRUPTS FROM TRASHING THE STACK IF A
5750          ;                               ;USER MODE HALT OCCURS.
5751 023430   012737 140340 177776          ;      MOV     #140340,#177776      ;SETUP PSW TO USER MODE
5752 023436   012737 160000 177572          ;      MOV     #160000,#177572      ;SETUP MMRO
5753 023444   012737 000077 172516          ;      MOV     #77,#172516          ;SETUP MMR3
5754 023452   012737 077000 177772          ;      MOV     #77000,#177772      ;SETUP PIRQ
5755 023460   000277          ;      SCC     ;SFT ALL CC BITS

```

***** DOUBLE OPERAND TESTS *****

```

5756 023462 000005          RESET
5757 023464 022737 140357 177776    CMP      #140357,#177776    ; TEST INSTRUCTION
5758 023472 001402          BEQ      5#                ; IS PSW CORRECT
5759 023474 000000          HALT          ; YES GO ON
5760 023476 104001          ERROR     +1                ; USER MODE HALT; WILL TRAP TO LOC 4
5761                                ; CPU ERROR
5762 023500 013701 177572 5#:      MOV      #177572,R1        ; NO GO TO ERROR
5763 023504 042701 000176          BIC      #176,R1          ; SAVE MMRO IN R1
5764 023510 022701 160000          CMP      #160000,R1      ; CLEAR BITS 1-6 FROM MMRO
5765 023514 001402          BEQ      6#                ; IS MMRO CORRECT
5766 023516 000000          HALT          ; YES GO ON
5767 023520 104001          ERROR     +1                ; USER MODE HALT; WILL TRAP TO LOC 4
5768                                ; CPU ERROR
5769 023522 022737 000077 172516 6#:  CMP      #77,#172516      ; NO GO TO ERROR
5770 023530 001402          BEQ      7#                ; IS MMR3 CORRECT
5771 023532 000000          HALT          ; YES GO ON
5772 023534 104001          ERROR     +1                ; USER MODE HALT; WILL TRAP TO LOC 4
5773                                ; CPU ERROR
5774 023536 022737 077314 177772 7#:  CMP      #77314,#177772  ; NO GO TO ERROR
5775 023544 001403          BEQ      9#                ; IS PIRQ CORRECT
5776 023546 000000          HALT          ; YES GO ON
5777 023550 104001          ERROR     +1                ; USER MODE HALT; WILL TRAP TO LOC 4
5778                                ; CPU ERROR
5779 023552 000002          RTI          ; NO GO TO ERROR
5780                                ; USER MODE HALT OCCURRED; GO TO ERROR.
5781 023554 005037 177772          9#:      CLR      #177772      ; CLEAR PIRQ
5782 023560 005037 177776          CLR      #177776      ; CLEAR PSW
5783 023564 010237 000004          MOV      R2,#4          ; RESTORE VECTORS TO PREVIOUS STATE
5784 023570 010337 000006          MOV      R3,#6          ;
5785 023574          FIN121:
5786                                ;
5789 023574          TE122:
5790                                ;
5791                                ; TEST SPL (SET PRIORITY LEVEL)
5792 023574 012705 000010          MOV      #8.,R5        ; INIT COUNTER
5793 023600 012701 024024          MOV      #T122B,R1     ; SETUP POINTER TO DATA
5794 023604 012737 030000 177776 1#:  MOV      #30000,#177776 ; INIT PSW
5795 023612 004767 000054          JSR      PC,T122A      ; TEST INSTRUCTION
5796 023616 022137 177776          CMP      (R1)+,#177776 ; IS PSW CORRECT
5797 023622 001401          BEQ      2#                ; YES GO ON
5798 023624 104001          ERROR     +1                ; CPU ERROR
5799                                ; NO GO TO ERROR
5800 023626 077512          2#:      SOB      R5,1#        ; REPEAT UNTIL ALL CASES ARE TESTED
5801
5802
5803 023630 012705 000010          MOV      #8.,R5        ; INIT COUNTER
5804 023634 012737 140000 177776 3#:  MOV      #140000,#177776 ; SETUP PSW TO USER MODE
5805 023642 012706 000600          MOV      #600,R6       ; SETUP USER STACK
5806 023646 004767 000020          JSR      PC,T122A      ; TEST INSTRUCTION
5807 023652 022737 140017 177776          CMP      #140017,#177776 ; IS PSW CORRECT
5808 023660 001401          BEQ      4#                ; YES GO ON
5809 023662 104001          ERROR     +1                ; CPU ERROR
5810                                ; NO GO TO ERROR
5811 023664 077515          4#:      SOB      R5,3#        ; REPEAT UNTIL ALL CASES ARE TESTED
5812
5813
5814 023666 000167 000152          JMP      FIN122

```


***** DOUBLE OPERAND TESTS *****

```

5815
5816 023672 020527 000010      ; T122A:  CMP      R5,#8.      ;FIND OUT WHAT COUNTER IS
5817 023676 001003              BNE      1$                ;IF NOT PRIORITY 0 GO TO 1$
5818 023700 000277              SCC                      ;SET ALL CC BITS
5819 023702 000230              SPL      0                ;SET PRIORITY TO 0
5820 023704 000446              BR       8$                ;RETURN
5821 023706 020527 000007      1$:    CMP      R5,#7        ;FIND OUT WHAT COUNTER IS
5822 023712 001003              BNE      2$                ;IF NOT PRIORITY 1 GO TO 2$
5823 023714 000277              SCC                      ;SET ALL CC BITS
5824 023716 000231              SPL      1                ;SET PRIORITY TO 1
5825 023720 000440              BR       8$                ;RETURN
5826 023722 020527 000006      2$:    CMP      R5,#6        ;FIND OUT WHAT COUNTER IS
5827 023726 001003              BNE      3$                ;IF NOT PRIORITY 2 GO TO 3$
5828 023730 000277              SCC                      ;SET ALL CC BITS
5829 023732 000232              SPL      2                ;SET PRIORITY TO 2
5830 023734 000432              BR       4$                ;RETURN
5831 023736 020527 000005      3$:    CMP      R5,#5        ;FIND OUT WHAT COUNTER IS
5832 023742 001003              BNE      4$                ;IF NOT PRIORITY 3 GO TO 4$
5833 023744 000277              SCC                      ;SET ALL CC BITS
5834 023746 000233              SPL      3                ;SET PRIORITY TO 3
5835 023750 000424              BR       8$                ;RETURN
5836 023752 020527 000004      4$:    CMP      R5,#4        ;FIND OUT WHAT COUNTER IS
5837 023756 001003              BNE      5$                ;IF NOT PRIORITY 4 GO TO 5$
5838 023760 000277              SCC                      ;SET ALL CC BITS
5839 023762 000234              SPL      4                ;SET PRIORITY TO 4
5840 023764 000416              BR       8$                ;RETURN
5841 023766 020527 000003      5$:    CMP      R5,#3        ;FIND OUT WHAT COUNTER IS
5842 023772 001003              BNE      6$                ;IF NOT PRIORITY 5 GO TO 6$
5843 023774 000277              SCC                      ;SET ALL CC BITS
5844 023776 000235              SPL      5                ;SET PRIORITY TO 5
5845 024000 000410              BR       8$                ;RETURN
5846 024002 020527 000002      6$:    CMP      R5,#2        ;FIND OUT WHAT COUNTER IS
5847 024006 001003              BNE      7$                ;IF NOT PRIORITY 6 GO TO 7$
5848 024010 000277              SCC                      ;SET ALL CC BITS
5849 024012 000236              SPL      6                ;SET PRIORITY TO 6
5850 024014 000402              BR       8$                ;RETURN
5851 024016 000277              7$:    SCC                      ;SET ALL CC BITS
5852 024020 000237              SPL      7                ;SET PRIORITY TO 7
5853 024022 000207              8$:    RTS      PC          ;RETURN
5854
5855 024024 030017      ; T122B:  .WORD      30017
5856 024026 030057      .WORD      30057
5857 024030 030117      .WORD      30117
5858 024032 030157      .WORD      30157
5859 024034 030217      .WORD      30217
5860 024036 030257      .WORD      30257
5861 024040 030317      .WORD      30317
5862 024042 030357      .WORD      30357
5863 024044 000240      FIN122:  NOP
5864
5867 024046      ; TE123:
5868      ;
5869 024046 005037 177776      ; TEST TSTSET INSTRUCTION (MULTI PROCESSING INST)
5870 024052 012703 000012      CLR      0#177776        ;INIT PSW
5871 024056 012701 000400      MOV      0#10.,R3        ;INIT COUNTER
5872 024062 012700 024230      MOV      0#400,R1        ;SETUP DESTINATION
5873 024066 012021      MOV      0#T123A,R0      ;SETUP SOURCE
100$:  MOV      (R0)+,(R1)+  ;RELOCATE TABLES

```

***** DOUBLE OPERAND TESTS *****

```

5874 024070 077302 SOB R3,100$ ;ARE WE DONE
5875 024072 013746 000010 MOV @#10,-(SP) ;SAVE VECTOR
5876 024076 012737 024254 000010 MOV @T123D,@#10 ;SETUP NEW VECTOR
5877 024104 005000 CLR R0 ;INIT R0
5878 024106 012701 000400 MOV @400,R1 ;SETUP POINTERS TO TABLES
5879 024112 012702 000410 MOV @410,R2 ;
5880 024116 012703 000416 MOV @416,R3 ;
5881 024122 010104 MOV R1,R4 ;
5882 024124 012737 030000 177776 1$: MOV @30000,@#177776 ;SETUP PSW
5883 024132 000262 SEV ;SET V BIT
5884 024134 007221 .WORD 7221 ; TEST INSTRUCTION
5885 024136 022237 177776 CMP (R2)+,@#177776 ;IS PSW CORRECT
5886 024142 001401 BEQ 2$ ;YES GO ON
5887 024144 104001 ERROR +1 ;CPU ERROR
5888 ;NO GO TO ERROR
5889 024146 020013 2$: CMP R0,(R3) ;IS R0 CORRECT
5890 024150 001401 BEQ 3$ ;YES GO ON
5891 024152 104001 ERROR +1 ;CPU ERROR
5892 ;NO GO TO ERROR
5893 024154 005204 3$: INC R4 ;SETUP EXPECTED DATA
5894 024156 005204 INC R4 ;
5895 024160 020401 CMP R4,R1 ;IS R1 CORRECT
5896 024162 001401 BEQ 4$ ;YES GO ON
5897 024164 104001 ERROR +1 ;CPU ERROR
5898 ;NO GO TO ERROR
5899 024166 052713 000001 4$: BIS @1,(R3) ;SETUP EXPECTED DATA
5900 024172 022341 CMP (R3)+,-(R1) ;IS TEST LOCATION CORRECT
5901 024174 001401 BEQ 5$ ;YES GO ON
5902 024176 104001 ERROR +1 ;CPU ERROR
5903 ;NO GO TO ERROR
5904 024200 005201 5$: INC R1 ;POINT TO NEXT TEST LOCATION
5905 024202 005201 INC R1 ;
5906 024204 021127 177777 CMP (R1),@#177777 ;ARE WE DONE
5907 024210 001345 BNE 1$ ;NO GO TO 1$
5908 024212 012737 024256 000010 MOV @T123E,@#10 ;SETUP NEW VECTOR
5909 024220 007201 .WORD 7201 ; TEST INSTRUCTION ILLEGAL MODE
5910 024222 104001 ERROR +1 ;CPU ERROR
5911 ;GO TO ERROR IF DIDN'T TRAP
5912 024224 000167 000032 JMP T123F
5913 ;
5914 ;
5915 024230 167604 T123A: .WORD 167604
5916 024232 000000 .WORD 0
5917 024234 000001 .WORD 1
5918 024236 177777 .WORD 177777
5919 024240 030010 T123B: .WORD 30010
5920 024242 030004 .WORD 30004
5921 024244 030001 .WORD 30001
5922 024246 167604 T123C: .WORD 167604
5923 024250 000000 .WORD 0
5924 024252 000001 .WORD 1
5925 024254 104001 T123D: ERROR +1 ;CPU ERROR
5926 ;GO TO ERROR IF TRAPPED
5927 024256 005726 T123E: TST (SP)+ ;CLEAN UP STACK
5928 024260 005726 TST (SP)+ ;
5929 024262 012637 000010 T123F: MOV (SP)+,@#10 ;RESTORE VECTOR
5930

```

***** DOUBLE OPERAND TESTS *****

```

5931
5934 024266      ;
5935      ;      TE124:
5936 024266 005037 177776      ;      TEST WRTLCK (WRITE LOCK MULTI PROCESSING INST)
5937 024272 012703 000012      CLR      @#177776      ;INIT PSW
5938 024276 012701 000400      MOV      @10.,R3      ;INIT COUNTER
5939 024302 012700 024462      MOV      @400,R1      ;SETUP DESTINATION
5940 024306 012021      100$: MOV      @T124A,R0      ;SETUP SOURCE
5941 024310 077302      SOB      R3,100$      ;RELOCATE TABLES
5942 024312 013746 000010      MOV      @#10,-(SP)    ;ARE WE DONE
5943 024316 012737 024506 000010 MOV      @T124D,@#10    ;SAVE VECTOR
5944 024324 012701 000400      MOV      @400,R1      ;SETUP NEW VECTOR
5945 024330 012702 000410      MOV      @410,R2      ;SETUP POINTERS TO TABLES
5946 024334 012703 000416      MOV      @416,R3      ;
5947 024340 010204      MOV      R2,R4      ;
5948 024342 012737 030000 177776 1$: MOV      @30000,@#177776 ;SETUP PSW
5949 024350 011100      MOV      (R1),R0      ;SETUP R0
5950 024352 020327 000416      CMP      R3,@416      ;IS THIS THE FIRST TEST CASE
5951 024356 001401      BEQ      2$          ;YES GO TO 2$
5952 024360 000402      BR       3$          ;NO GO TO 3$
5953 024362 000261      2$: SEC          ;SET C BIT
5954 024364 000401      BR       4$          ;
5955 024366 000241      3$: CLC          ;CLEAR C BIT
5956 024370 000262      4$: SEV          ;SET V BIT
5957 024372 007322      .WORD    7322        ; TEST INSTRUCTION
5958 024374 022337 177776      CMP      (R3)+,@#177776 ;IS PSW CORRECT
5959 024400 001401      BEQ      5$          ;YES GO ON
5960 024402 104001      ERROR    +1          ;CPU ERROR
5961      ;NO GO TO ERROR
5962 024404 021100      5$: CMP      (R1),R0      ;IS R0 CORRECT
5963 024406 001401      BEQ      6$          ;YES GO ON
5964 024410 104001      ERROR    +1          ;CPU ERROR
5965      ;NO GO TO ERROR
5966 024412 005204      6$: INC      R4          ;SETUP EXPECTED DATA
5967 024414 005204      INC      R4          ;
5968 024416 020204      CMP      R2,R4      ;IS R2 CORRECT
5969 024420 001401      BEQ      7$          ;YES GO ON
5970 024422 104001      ERROR    +1          ;CPU ERROR
5971      ;NO GO TO ERROR
5972 024424 022142      7$: CMP      (R1)+,-(R2)  ;IS TEST LOCATION CORRECT
5973 024426 001401      BEQ      8$          ;YES GO ON
5974 024430 104001      ERROR    +1          ;CPU ERROR
5975      ;NO GO TO ERROR
5976 024432 005202      8$: INC      R2          ;POINT TO NEXT TEST LOCATION
5977 024434 005202      INC      R2          ;
5978 024436 021127 177777      CMP      (R1),@177777 ;ARE WE DONE
5979 024442 001337      BNE      1$          ;NO GO TO 1$
5980 024444 012737 024510 000010 MOV      @T124E,@#10    ;SETUP NEW VECTOR
5981 024452 007302      .WORD    7302        ; TEST INSTRUCTION ILLEGAL MODE
5982 024454 104001      ERROR    +1          ;CPU ERROR
5983      ;GO TO ERROR IF DIDN'T TRAP
5984 024456 000167 000032      JMP      T124F
5985      ;
5986      ;
5987 024462 167604      T124A: .WORD    167604
5988 024464 000000      .WORD    0
5989 024466 000001      .WORD    1

```

***** DOUBLE OPERAND TESTS *****

```

5990 024470 177777          .WORD 177777
5991 024472 177777          .WORD 177777
5992 024474 177777          .WORD 177777
5993 024476 177777          .WORD 177777
5994 024500 030011          .WORD 30011
5995 024502 030004          .WORD 30004
5996 024504 030000          .WORD 30000
5997 024506 104001          T124D: ERROR +1
5998
5999 024510 005726          T124E: TST (SP)+
6000 024512 005726          TST (SP)+
6001 024514 012637 000010  T124F: MOV (SP)+, #10
6002
6003
6006 024520          ;
6007          ;
6008 024520 005037 177776          ; TEST MUL (MULTIPLY INST)
6009 024524 012701 024760          CLR #177776 ;INIT PS
6010          MOV #TE125A,R1 ;SETUP POINTERS TO TABLES
6011 024530 010137 114146          1$: MOV R1, #EXPDAT
6012 024534 062737 000002 114146 ADD #2, #EXPDAT
6013 024542 012703 122222          MOV #122222,R3 ;POINT TO SOURCE
6014 024546 011102          MOV (R1),R2 ;INIT R3 TO A KNOWN STATE
6015 024550 000277          SCC ;INIT DESTINATION REG
6016 024552 070261 000002          MUL 2(R1),R2 ;SET ALL CC BITS
6017 024556 026137 000004 177776 CMP 4(R1), #177776 ; TEST INSTRUCTION
6018 024564 001401          BEQ 2$ ;IS PS CORRECT
6019 024566 104001          ERROR +1 ;YES GO ON
6020          ;CPU ERROR
6021 024570 026103 000006          2$: CMP 6(R1),R3 ;NO GO TO ERROR
6022 024574 001401          BEQ 3$ ;IS R3 CORRECT
6023 024576 104001          ERROR +1 ;YES GO ON
6024          ;CPU ERROR
6025 024600 026102 000010          3$: CMP 10(R1),R2 ;NO GO TO ERROR
6026 024604 001401          BEQ 4$ ;IS R2 CORRECT
6027 024606 104001          ERROR +1 ;YES GO ON
6028          ;CPU ERROR
6029 024610 026177 000002 067330 4$: CMP 2(R1), #EXPDAT ;NO GO TO ERROR
6030 024616 001401          BEQ 5$ ;IS SOURCE LOCATION OK
6031 024620 104001          ERROR +1 ;YES GO ON
6032          ;CPU ERROR
6033 024622 062701 000012          5$: ADD #12,R1 ;NO GO TO ERROR
6034 024626 020127 025206          CMP R1, #FIN125 ;GO TO NEXT TEST
6035 024632 001336          BNE 1$ ;ARE WE FINISHED
6036          ;NO GO TO 1$
6037          ;
6038          ;SECOND PART
6039          ;USING ODD REGISTER
6040          ;
6041 024634 012701 024760          6$: MOV #TE125A,R1 ;SETUP POINTERS TO TABLES
6042 024640          7$:
6043 024640 010102          MOV R1,R2
6044 024642 012706 001000          MOV #STBOT,R6 ;INIT R6 TO A KNOWN STATE
6045 024646 012704 000004          MOV #4,R4 ;SETUP R4 VALUE
6046 024652 011105          MOV (R1),R5 ;INIT DESTINATION REG
6047 024654 000277          SCC ;SET ALL CC BITS
6048 024656 070561 000002          MUL 2(R1),R5 ; TEST INSTRUCTION

```

***** DOUBLE OPERAND TESTS *****

```

6049 024662 026137 000004 177776      CMP      4(R1),0#177776      ;IS P5 CORRECT
6050 024670 001401                    BEQ      8$                  ;YES GO ON
6051 024672 104001                    ERROR   +1                  ;CPU ERROR
6052                                     ;NO GO TO ERROR
6053 024674 026105 000006      8$:    CMP      6(R1),R5      ;IS R5 CORRECT
6054 024700 001401                    BEQ      9$                  ;YES GO ON
6055 024702 104001                    ERROR   +1                  ;CPU ERROR
6056                                     ;NO GO TO ERROR
6057 024704 020627 001000      9$:    CMP      R6,#STBOT    ;IS R6 CORRECT
6058 024710 001403                    BEQ     10$                  ;YES GO ON
6059 024712 012706 001000      MOV     #STBOT,R6          ;RESTORE SP
6060 024716 104001                    ERROR   +1                  ;CPU ERROR
6061                                     ;NO GO TO ERROR
6062 024720 005722      10$:   TST     (R2)+          ;POINT TO SOURCE OPERAND
6063 024722 021261 000002      CMP     (R2),2(R1)        ;IS SOURCE LOCATION OK
6064 024726 001401                    BEQ     11$                  ;YES GO ON
6065 024730 104001                    ERROR   +1                  ;CPU ERROR
6066                                     ;NO GO TO ERROR
6067 024732 020427 000004      11$:   CMP     R4,#4          ;IS R4 CORRECT
6068 024736 001401                    BEQ     12$                  ;YES GO ON
6069 024740 104001                    ERROR   +1                  ;CPU ERROR
6070                                     ;NO GO TO ERROR
6071 024742 062701 000012      12$:   ADD     #12,R1         ;
6072 024746 020127 025206      CMP     R1,#FIN125        ;ARE WE FINISHED
6073 024752 001332                    BNE     7$                  ;NO GO TO 7$
6074
6075
6076 024754 000167 000226      JMP     FIN125
6077
6078
6079 024760 177777      ;
6080 024762 177777      ;TE125A: .WORD 177777      ;MULTIPLICAND
6081 024764 000000      .WORD 177777      ;MULTIPLIER
6082 024766 000001      .WORD 0
6083 024770 000000      .WORD 0
6084
6085 024772 006772      .WORD 0
6086 024774 100000      .WORD 6772          ;MULTIPLICAND
6087 024776 000011      .WORD 100000       ;MULTIPLIER
6088 025000 000000      .WORD 11
6089 025002 174403      .WORD 0
6090                                     .WORD 174403
6091 025004 177777      .WORD 177777       ;MULTIPLICAND
6092 025006 077777      .WORD 77777        ;MULTIPLIER
6093 025010 000010      .WORD 10
6094 025012 100001      .WORD 100001
6095 025014 177777      .WORD 177777
6096
6097 025016 077777      .WORD 77777        ;MULTIPLICAND
6098 025020 000456      .WORD 456          ;MULTIPLIER
6099 025022 000001      .WORD 1
6100 025024 177322      .WORD 177322
6101 025026 000226      .WORD 226
6102
6103 025030 173210      .WORD 173210       ;MULTIPLICAND
6104 025032 000000      .WORD 0            ;MULTIPLIER
6105 025034 000004      .WORD 4

```

***** DOUBLE OPERAND TESTS *****

6106	025036	000000	.WORD	0	
6107	025040	000000	.WORD	0	
6108					
6109	025042	000000	.WORD	0	;MULTIPLICAND
6110	025044	003251	.WORD	3251	;MULTIPLIER
6111	025046	000004	.WORD	4	
6112	025050	000000	.WORD	0	
6113	025052	000000	.WORD	0	
6114					
6115	025054	000000	.WORD	0	;MULTIPLICAND
6116	025056	000000	.WORD	0	;MULTIPLIER
6117	025060	000004	.WORD	4	
6118	025062	000000	.WORD	0	
6119	025064	000000	.WORD	0	
6120					
6121	025066	100000	.WORD	100000	;MULTIPLICAND
6122	025070	000001	.WORD	1	;MULTIPLIER
6123	025072	000010	.WORD	10	
6124	025074	100000	.WORD	100000	
6125	025076	177777	.WORD	177777	
6126					
6127	025100	077777	.WORD	77777	;MULTIPLICAND
6128	025102	000001	.WORD	1	;MULTIPLIER
6129	025104	000000	.WORD	0	
6130	025106	077777	.WORD	77777	
6131	025110	000000	.WORD	0	
6132					
6133	025112	000010	.WORD	10	;MULTIPLICAND
6134	025114	010000	.WORD	10000	;MULTIPLIER
6135	025116	000001	.WORD	1	
6136	025120	100000	.WORD	100000	
6137	025122	000000	.WORD	0	
6138					
6139	025124	001452	.WORD	1452	;MULTIPLICAND
6140	025126	034527	.WORD	34527	;MULTIPLIER
6141	025130	000001	.WORD	1	
6142	025132	066506	.WORD	66506	
6143	025134	000265	.WORD	265	
6144					
6145	025136	000007	.WORD	7	;MULTIPLICAND
6146	025140	000400	.WORD	400	;MULTIPLIER
6147	025142	000000	.WORD	0	
6148	025144	003400	.WORD	3400	
6149	025146	000000	.WORD	0	
6150					
6151	025150	000002	.WORD	2	;MULTIPLICAND
6152	025152	100000	.WORD	100000	;MULTIPLIER
6153	025154	000011	.WORD	11	
6154	025156	000000	.WORD	0	
6155	025160	177777	.WORD	177777	
6156					
6157	025162	100000	.WORD	100000	;MULTIPLICAND
6158	025164	077777	.WORD	77777	;MULTIPLIER
6159	025166	000011	.WORD	11	
6160	025170	100000	.WORD	100000	
6161	025172	140000	.WORD	140000	
6162					

***** DOUBLE OPERAND TESTS *****

```

6163 025174 000001          .WORD 1          ;MULTIPLICAND
6164 025176 177777          .WORD 177777     ;MULTIPLIER
6165 025200 000010          WORD 10
6166 025202 177777          .WORD 177777
6167 025204 177777          .WORD 177777
6168 025206          FIN125:
6169          ;
6172 025206          TE126:
6173          ;
6174 025206 005037 177776          CLR 00177776     ;INIT PSW
6175 025212 005006          CLR R6           ;INIT SP
6176 025214 013705 000000          MOV 000,R5      ;SAVE VECTORS
6177 025220 013701 000002          MOV 002,R1      ;
6178 025224 012737 000137 000000          MOV 0137,000   ;SETUP NEW VECTORS
6179 025232 012737 025254 000002          MOV 0TE126A,002 ;
6180 025240 000277          SCC             ;SET ALL CC BITS
6181 025242 071627 000002          DIV 02,R6       ; TEST INSTRUCTION
6182 025246 012706 001000          A126: MOV 0STBOT,R6 ;RESTORE SP BEFORE GOING TO ERROR
6183 025252 104001          ERROR +1       ;CPU ERROR
6184          ;IF R7 ISN'T CORRECT GO TO ERROR
6185 025254 022737 000000 177776 TE126A: CMP 00,00177776 ;IS PS CORRECT
6186 025262 001403          BEQ 1$          ;YES GO ON
6187 025264 012706 001000          MOV 0STBOT,R6  ;RESTORE SP BEFORE GOING TO ERROR
6188 025270 104001          ERROR +1       ;CPU ERROR
6189          ;NO GO TO ERROR
6190 025272 012704 025246          1$: MOV 0A126,R4   ;SETUP EXPECTED DATA
6191 025276 006204          ASR R4         ;
6192 025300 020406          CMP R4,R6      ;IS R6 CORRECT
6193 025302 001403          BEQ 2$          ;YES GO ON
6194 025304 012706 001000          MOV 0STBOT,R6  ;RESTORE SP BEFORE GOING TO ERROR
6195 025310 104001          ERROR +1       ;CPU ERROR
6196          ;NO GO TO ERROR
6197 025312 010537 000000          2$: MOV R5,000   ;RESTORE VECTORS
6198 025316 010137 000002          MOV R1,002     ;
6199 025322 012706 001000          MOV 0STBOT,R6  ;INIT SP
6200 025326 012702 000006          MOV 06,R2      ;INIT GPR 2
6201 025332 012703 000047          MOV 047,R3     ;INIT GPR 3
6202 025336 000277          SCC             ;SET ALL CC BITS
6203 025340 071302          DIV R2,R3     ; TEST INSTRUCTION
6204 025342 022737 000002 177776          CMP 02,00177776 ;IS PS CORRECT
6205 025350 001401          BEQ 3$          ;YES GO ON
6206 025352 104001          ERROR +1       ;CPU ERROR
6207          ;NO GO TO ERROR
6208 025354 022702 000006          3$: CMP 06,R2     ;IS R2 CORRECT
6209 025360 001401          BEQ 4$          ;YES GO ON
6210 025362 104001          ERROR +1       ;CPU ERROR
6211          ;NO GO TO ERROR
6212 025364 022703 000047          4$: CMP 047,R3   ;IS R3 CORRECT
6213 025370 001401          BEQ 5$          ;YES GO ON
6214 025372 104001          ERROR +1       ;CPU ERROR
6215          ;NO GO TO ERROR
6216 025374 005004          5$: CLR R4        ;INIT R4
6217 025376 012705 000004          MOV 04,R5      ;INIT R5
6218 025402 000277          SCC             ;SET ALL CC BITS
6219 025404 071427 000000          DIV 00,R4     ; TEST INSTRUCTION
6220 025410 022737 000007 177776          CMP 07,00177776 ;IS PS CORRECT
6221 025416 001401          BEQ 6$          ;YES GO ON

```

***** DOUBLE OPERAND TESTS *****

6222	025420	104001			ERROR	+1			;CPU ERROR
6223									;NO GO TO ERROR
6224	025422	022704	000000	6#:	CMP	#0,R4			;IS R4 CORRECT
6225	025426	001401			BEQ	7#			;YES GO ON
6226	025430	104001			ERROR	+1			;CPU ERROR
6227									;NO GO TO ERROR
6228	025432	022705	000004	7#:	CMP	#4,R5			;IS R5 CORRECT
6229	025436	001401			BEQ	8#			;YES GO ON
6230	025440	104001			ERROR	+1			;CPU ERROR
6231									;NO GO TO ERROR
6232	025442	012700	000004	8#:	MOV	#4,R0			;INIT R0
6233	025446	012705	000010		MOV	#10,R5			;INIT R5
6234	025452	005004			CLR	R4			;INIT R4
6235	025454	000277			SCC				;SET ALL CC BITS
6236	025456	071400			DIV	R0,R4			; TEST INSTRUCTION
6237	025460	022737	000000	177776	CMP	#0,#177776			;IS PS CORRECT
6238	025466	001405			BEQ	9#			;YES GO ON
6239	025470	010067	152704		MOV	R0,400			;SAVE R0
6240	025474	104001			ERROR	+1			;CPU ERROR
6241									;NO GO TO ERROR
6242	025476	016700	152676		MOV	400,R0			;RESTORE R0
6243	025502	022700	000004	9#:	CMP	#4,R0			;IS R0 CORRECT
6244	025506	001401			BEQ	10#			;YES GO ON
6245	025510	104001			ERROR	+1			;CPU ERROR
6246									;NO GO TO ERROR
6247	025512	022704	000002	10#:	CMP	#2,R4			;IS R4 CORRECT
6248	025516	001401			BEQ	11#			;YES GO ON
6249	025520	104001			ERROR	+1			;CPU ERROR
6250									;NO GO TO ERROR
6251	025522	022705	000000	11#:	CMP	#0,R5			;IS R5 CORRECT
6252	025526	001401			BEQ	12#			;YES GO ON
6253	025530	104001			ERROR	+1			;CPU ERROR
6254									;NO GO TO ERROR
6255	025532	012705	000010	12#:	MOV	#10,R5			;INIT R5
6256	025536	005004			CLR	R4			;INIT R4
6257	025540	000277			SCC				;SET ALL CC BITS
6258	025542	071427	000003		DIV	#3,R4			; TEST INSTRUCTION
6259	025546	022737	000000	177776	CMP	#0,#177776			;IS PS CORRECT
6260	025554	001401			BEQ	13#			;YES GO ON
6261	025556	104001			ERROR	+1			;CPU ERROR
6262									;NO GO TO ERROR
6263	025560	022704	000002	13#:	CMP	#2,R4			;IS R4 CORRECT
6264	025564	001401			BEQ	14#			;YES GO ON
6265	025566	104001			ERROR	+1			;CPU ERROR
6266									;NO GO TO ERROR
6267	025570	022705	000002	14#:	CMP	#2,R5			;IS R5 CORRECT
6268	025574	001401			BEQ	15#			;YES GO ON
6269	025576	104001			ERROR	+1			;CPU ERROR
6270									;NO GO TO ERROR
6271				:					
6272				:					
6273				:					
6274	025600	012701	025730	15#:	MOV	#TE1268,R1			;SETUP POINTERS TO TABLES
6275									
6276	025604	010137	114146	16#:	MOV	R1,#EXPDAT			;SAVE A COPY OF R1
6277	025610	011104			MOV	(R1),R4			;INIT R4
6278	025612	016103	000004		MOV	4(R1),R3			;SAVE SOURCE

***** ** DOUBLE OPERAND TESTS *****

```

6279
6280 025616 016105 000002      ;      MOV      2(R1),R5      ;INIT R5
6281 025622 000277      SCC      ;SET ALL CC BITS
6282 025624 071461 000004      DIV      4(R1),R4      ; TEST INSTRUCTION
6283 025630 026137 000006 177776  CMP      6(R1),0#177776 ;IS PS CORRECT
6284 025636 001401      BEQ      17#      ;YES GO ON
6285 025640 104001      ERROR   +1      ;CPU ERROR
6286      ;NO GO TO ERROR
6287 025642 026105 000010 17# :  CMP      10(R1),R5      ;IS R5 CORRECT
6288 025646 001401      BEQ      18#      ;YES GO ON
6289 025650 104001      ERROR   +1      ;CPU ERROR
6290      ;NO GO TO ERROR
6291 025652 026104 000012 18# :  CMP      12(R1),R4      ;IS R4 CORRECT
6292 025656 001401      BEQ      19#      ;YES GO ON
6293 025660 104001      ERROR   +1      ;CPU ERROR
6294      ;NO GO TO ERROR
6295 025662 023701 114'46 19# :  CMP      0#EXPDAT,R1      ;IS R1 CORRECT
6296 025666 001403      BEQ      20#      ;YES GO ON
6297 025670 104001      ERROR   +1      ;CPU ERROR
6298      ;NO GO TO ERROR
6299 025672 013701 114146 20# :  MOV      0#EXPDAT,R1      ;RESTORE CORRECT VALUE
6300 025676 026103 000004      CMP      4(R1),R3      ;IS SOURCE CORRECT
6301 025702 001403      BEQ      21#      ;YES GO ON
6302 025704 104001      ERROR   +1      ;CPU ERROR
6303      ;NO GO TO ERROR
6304 025706 010361 000004 21# :  MOV      R3,4(R1)      ;TRY TO RESTORE CODE
6305 025712 062701 000014      ADD      #14,R1      ;POINT TO NEXT LOCATION
6306 025716 021127 000333      CMP      (R1),#333      ;ARE WE DONE
6307 025722 001330      BNE     16#      ;NO GO TO 16#
6308
6309
6310 025724 000167 000316      JMP      FIN126
6311      ;
6312      ;
6313 025730 177777      TE126B: .WORD 177777 ;DIVIDEND
6314 025732 177777      .WORD 177777 ;INIT R5
6315 025734 177777      .WORD 177777 ;DIVISOR
6316 025736 000000      .WORD 0 ;PSW
6317 025740 000000      .WORD 0 ;R5 RESULT
6318 025742 000001      .WORD 1 ;R4 RESULT
6319
6320 025744 000000      .WORD 0 ;DIVIDEND
6321 025746 177777      .WORD 177777 ;INIT R5
6322 025750 177777      .WORD 177777 ;DIVISOR
6323 025752 000012      .WORD 12 ;PSW
6324 025754 177777      .WORD 177777 ;R5 RESULT
6325 025756 000000      .WORD 0 ;R4 RESULT
6326
6327 025760 177777      .WORD 177777 ;DIVIDEND
6328 025762 000000      .WORD 0 ;INIT R5
6329 025764 177777      .WORD 177777 ;DIVISOR
6330 025766 000002      .WORD 2 ;PSW
6331 025770 000000      .WORD 0 ;R5 RESULT
6332 025772 177777      .WORD 177777 ;R4 RESULT
6333
6334 025774 000000      .WORD 0 ;DIVIDEND
6335 025776 007642      .WORD 7642 ;INIT R5

```

***** DOUBLE OPERAND TESTS *****

6336	026000	007643	.WORD	7643	;DIVISOR
6337	026002	000004	.WORD	4	;PSW
6338	026004	007642	.WORD	7642	;R5 RESULT
6339	026006	000000	.WORD	0	;R4 RESULT
6340					
6341	026010	000000	.WORD	0	;DIVIDEND
6342	026012	000137	.WORD	137	;INIT R5
6343	026014	177543	.WORD	177543	;DIVISOR
6344	026016	000004	.WORD	4	;PSW
6345	026020	000137	.WORD	137	;R5 RESULT
6346	026022	000000	.WORD	0	;R4 RESULT
6347					
6348	026024	000000	.WORD	0	;DIVIDEND
6349	026026	007643	.WORD	7643	;INIT R5
6350	026030	007643	.WORD	7643	;DIVISOR
6351	026032	000000	.WORD	0	;PSW
6352	026034	000000	.WORD	0	;R5 RESULT
6353	026036	000001	.WORD	1	;R4 RESULT
6354					
6355	026040	100000	.WORD	100000	;DIVIDEND
6356	026042	004376	.WORD	4376	;INIT R5
6357	026044	010021	.WORD	10021	;DIVISOR
6358	026046	000012	.WORD	12	;PSW
6359	026050	004376	.WORD	4376	;R5 RESULT
6360	026052	100000	.WORD	100000	;R4 RESULT
6361					
6362	026054	177700	.WORD	177700	;DIVIDEND
6363	026056	170033	.WORD	170033	;INIT R5
6364	026060	010021	.WORD	10021	;DIVISOR
6365	026062	000010	.WORD	10	;PSW
6366	026064	171307	.WORD	171307	;R5 RESULT
6367	026066	176024	.WORD	176024	;R4 RESULT
6368					
6369	026070	177700	.WORD	177700	;DIVIDEND
6370	026072	170033	.WORD	170033	;INIT R5
6371	026074	167757	.WORD	167757	;DIVISOR
6372	026076	000000	.WORD	0	;PSW
6373	026100	171307	.WORD	171307	;R5 RESULT
6374	026102	001754	.WORD	1754	;R4 RESULT
6375					
6376	026104	000000	.WORD	0	;DIVIDEND
6377	026106	177777	.WORD	177777	;INIT R5
6378	026110	000001	.WORD	1	;DIVISOR
6379	026112	000002	.WORD	2	;PSW
6380	026114	177777	.WORD	177777	;R5 RESULT
6381	026116	000000	.WORD	0	;R4 RESULT
6382					
6383	026120	177777	.WORD	177777	;DIVIDEND
6384	026122	045716	.WORD	45716	;INIT R5
6385	026124	000001	.WORD	1	;DIVISOR
6386	026126	000012	.WORD	12	;PSW
6387	026130	045716	.WORD	45716	;R5 RESULT
6388	026132	177777	.WORD	177777	;R4 RESULT
6389					
6390	026134	000000	.WORD	0	;DIVIDEND
6391	026136	000002	.WORD	2	;INIT R5
6392	026140	177770	.WORD	177770	;DIVISOR

***** DOUBLE OPERAND TESTS *****

6393	026142	000004		.WORD	4		;PSW
6394	026144	000002		.WORD	2		;R5 RESULT
6395	026146	000000		.WORD	0		;R4 RESULT
6396							
6397	026150	177777		.WORD	177777		;DIVIDEND
6398	026152	177776		.WORD	177776		;INIT R5
6399	026154	000010		.WORD	10		;DIVISOR
6400	026156	000004		.WORD	4		;PSW
6401	026160	177776		.WORD	177776		;R5 RESULT
6402	026162	000000		.WORD	0		;R4 RESULT
6403							
6404	026164	000001		.WORD	1		;DIVIDEND
6405	026166	177777		.WORD	177777		;INIT R5
6406	026170	000001		.WORD	1		;DIVISOR
6407	026172	000002		.WORD	2		;PSW
6408	026174	177777		.WORD	177777		;R5 RESULT
6409	026176	000001		.WORD	1		;R4 RESULT
6410							
6411	026200	000001		.WORD	1		;DIVIDEND
6412	026202	000000		.WORD	0		;INIT R5
6413	026204	000002		.WORD	2		;DIVISOR
6414	026206	000002		.WORD	2		;PSW
6415	026210	000000		.WORD	0		;R5 RESULT
6416	026212	000001		.WORD	1		;R4 RESULT
6417							
6418	026214	000001		.WORD	1		;DIVIDEND
6419	026216	000000		.WORD	0		;INIT R5
6420	026220	000003		.WORD	3		;DIVISOR
6421	026222	000000		.WORD	0		;PSW
6422	026224	000001		.WORD	1		;R5 RESULT
6423	026226	052525		.WORD	52525		;R4 RESULT
6424							
6425	026230	000023		.WORD	23		;DIVIDEND
6426	026232	016054		.WORD	16054		;INIT R5
6427	026234	016537		.WORD	16537		;DIVISOR
6428	026236	000000		.WORD	0		;PSW
6429	026240	010222		.WORD	10222		;R5 RESULT
6430	026242	000246		.WORD	246		;R4 RESULT
6431							
6432	026244	000333		.WORD	333		
6433	026246						
6434							
6437	026246						
6438							
6439	026246	005037	177776				
6440	026252	012702	000001				
6441	026256	000277					
6442	026260	072202					
6443	026262	022737	000000 177776				
6444	026270	001401					
6445	026272	104001					
6446							
6447	026274	020227	000002	1#:			
6448	026300	001401					
6449	026302	104001					
6450							
6451	026304	012702	100000	2#:			

FIN126:

TE127:

```

;
; TEST ASH (ARITHMETIC SHIFT)
CLR      #0177776          ;INIT PSW
MOV      #1,R2             ;SETUP OPERAND
SCC      ;SET ALL CC BITS
ASH      R2,R2             ; TEST INSTRUCTION
CMP      #0,#0177776      ;IS PS CORRECT
BEQ      1#                ;YES GO ON
ERROR    +1                ;CPU ERROR
;NO GO TO ERROR
CMP      R2,#2             ;IS R2 CORRECT
BEQ      2#                ;YES GO ON
ERROR    +1                ;CPU ERROR
;NO GO TO ERROR
MOV      #100000,R2       ;SETUP R2
    
```

***** DOUBLE OPERAND TESTS *****

```

6452 026310 012703 000001      MOV      #1,R3          ;SETUP R3
6453 026314 000257             CCC                ;CLEAR ALL CC BITS
6454 026316 072203             ASH      R3,R2        ; TEST INSTRUCTION
6455 026320 022737 000007 177776  CMP      #7,#0177776  ;IS PS CORRECT
6456 026326 001401             BEQ      3$           ;YES GO ON
6457 026330 104001             ERROR    +1          ;CPU ERROR
6458                                     ;NO GO TO ERROR
6459 026332 020327 000001      3$:    CMP      R3,#1    ;IS R3 CORRECT
6460 026336 001401             BEQ      4$           ;YES GO ON
6461 026340 104001             ERROR    +1          ;CPU ERROR
6462                                     ;NO GO TO ERROR
6463 026342 020227 000000      4$:    CMP      R2,#0    ;IS R2 CORRECT
6464 026346 001401             BEQ      5$           ;YES GO ON
6465 026350 104001             ERROR    +1          ;CPU ERROR
6466                                     ;NO GO TO ERROR
6467 026352 012701 026446      5$:    MOV      #TE127A,R1 ;SETUP POINTERS TO TABLES
6468
6469 026356 010103             MOV      R1,R3          ;
6470 026360 016102 000002      6$:    MOV      2(R1),R2    ;SETUP R2
6471 026364 000277             SCC                ;SET ALL CC BITS
6472 026366 072211             ASH      (R1),R2       ; TEST INSTRUCTION
6473 026370 026137 000004 177776  CMP      4(R1),#0177776 ;IS PS CORRECT
6474 026376 001401             BEQ      7$           ;YES GO ON
6475 026400 104001             ERROR    -1          ;CPU ERROR
6476                                     ;NO GO TO ERROR
6477 026402 026102 000006      7$:    CMP      6(R1),R2    ;IS R2 CORRECT
6478 026406 001401             BEQ      8$           ;YES GO ON
6479 026410 104001             ERROR    +1          ;CPU ERROR
6480                                     ;NO GO TO ERROR
6481 026412 020301             8$:    CMP      R3,R1    ;IS R1 CORRECT
6482 026414 001402             BEQ      9$           ;YES GO ON
6483 026416 104001             ERROR    +1          ;CPU ERROR
6484                                     ;NO GO TO ERROR
6485 026420 010301             MOV      R3,R1          ;RESTORE R1
6486 026422 021311             9$:    CMP      (R3),(R1) ;IS SOURCE CORRECT
6487 026424 001401             BEQ     10$           ;YES GO ON
6488 026426 104001             ERROR    +1          ;CPU ERROR
6489                                     ;NO GO TO ERROR
6490                                     ;SOURCE LOOKS INCORRECT
6491 026430 062701 000010      10$:   ADD      #10,R1    ;INCREMENT POINTER
6492 026434 020127 026706      CMP      R1,#FIN127  ;ARE WE DONE
6493 026440 001346             BNE     6$           ;NO GO TO 6$
6494
6495
6496 025442 000167 000240             JMP      FIN127
6497
6498
6499 026446 177761             ;
TE127A: .WORD 177761          ;SOURCE
6500 026450 077777             .WORD 77777          ;DEST
6501 026452 000005             .WORD 5
6502 026454 000000             .WORD 0
6503
6504 026456 177700             .WORD 177700        ;SOURCE
6505 026460 017777             .WORD 17777        ;DEST
6506 026462 000000             .WORD 0
6507 026464 017777             .WORD 17777
5508

```

***** DOUBLE OPERAND TESTS *****

6509	026466	177700	.WORD	177700	;SOURCE
6510	026470	100000	.WORD	100000	;DEST
6511	026472	000010	.WORD	10	
6512	026474	100000	.WORD	100000	
6513					
6514	026476	177777	.WORD	177777	;SOURCE
6515	026500	100000	.WORD	100000	;DEST
6516	026502	000010	.WORD	10	
6517	026504	140000	.WORD	140000	
6518					
6519	026506	177737	.WORD	177737	;SOURCE
6520	026510	177777	.WORD	177777	;DEST
6521	026512	000011	.WORD	11	
6522	026514	177777	.WORD	177777	
6523					
6524	026516	177706	.WORD	177706	;SOURCE
6525	026520	102000	.WORD	102000	;DEST
6526	026522	000007	.WORD	7	
6527	026524	000000	.WORD	0	
6528					
6529	026526	177710	.WORD	177710	;SOURCE
6530	026530	017777	.WORD	17777	;DEST
6531	026532	000013	.WORD	13	
6532	026534	177400	.WORD	177400	
6533					
6534	026536	177713	.WORD	177713	;SOURCE
6535	026540	000012	.WORD	12	;DEST
6536	026542	000000	.WORD	0	
6537	026544	050000	.WORD	50000	
6538					
6539	026546	177707	.WORD	177707	;SOURCE
6540	026550	170001	.WORD	170001	;DEST
6541	026552	000002	.WORD	2	
6542	026554	000200	.WORD	200	
6543					
6544	026556	177717	.WORD	177717	;SOURCE
6545	026560	000001	.WORD	1	;DEST
6546	026562	000012	.WORD	12	
6547	026564	100000	.WORD	100000	
6548					
6549	026566	177740	.WORD	177740	;SOURCE
6550	026570	017777	.WORD	17777	;DEST
6551	026572	000004	.WORD	4	
6552	026574	000000	.WORD	0	
6553					
6554	026576	177771	.WORD	177771	;SOURCE
6555	026600	150000	.WORD	150000	;DEST
6556	026602	000010	.WORD	10	
6557	026604	177640	.WORD	177640	
6558					
6559	026606	177742	.WORD	177742	;SOURCE
6560	026610	100000	.WORD	100000	;DEST
6561	026612	000011	.WORD	11	
6562	026614	177777	.WORD	177777	
6563					
6564	026616	177764	.WORD	177764	;SOURCE
6565	026620	100000	.WORD	100000	;DEST

***** DOUBLE OPERAND TESTS *****

6566	026622	000010		.WORD	10	
6567	026624	177770		.WORD	177770	
6568						
6569	026626	177750		.WORD	177750	;SOURCE
6570	026630	052525		.WORD	52525	;DEST
6571	026632	000004		.WORD	4	
6572	026634	000000		.WORD	0	
6573						
6574	026636	177760		.WORD	177760	;SOURCE
6575	026640	100000		.WORD	100000	;DEST
6576	026642	000011		.WORD	11	
6577	026644	177777		.WORD	177777	
6578						
6579	026646	177770		.WORD	177770	;SOURCE
6580	026650	100000		.WORD	100000	;DEST
6581	026652	000010		.WORD	10	
6582	026654	177600		.WORD	177600	
6583						
6584	026656	177712		.WORD	177712	;SOURCE
6585	026660	004367		.WORD	4367	;DEST
6586	026662	000013		.WORD	13	
6587	026664	156000		.WORD	156000	
6588						
6589	026666	177764		.WORD	177764	;SOURCE
6590	026670	017777		.WORD	17777	;DEST
6591	026672	000001		.WORD	1	
6592	026674	000001		.WORD	1	
6593						
6594	026676	177701		.WORD	177701	;SOURCE
6595	026700	110000		.WORD	110000	;DEST
6596	026702	000003		.WORD	3	
6597	026704	020000		.WORD	20000	
6598						
6599	026706	000240				
6600						
6603	026710					
6604						
6605	026710	005037	177776			
6606	026714	012701	000023			
6607	026720	012705	052525			
6608	026724	005004				
6609	026726	000277				
6610	026730	073401				
6611	026732	023727	177776 000012			
6612	026740	001401				
6613	026742	104001				
6614						
6615	026744	020127	000023	1\$:	CMP R1,#23	;IS R1 CORRECT
6616	026750	001401			BEQ 2\$;YES GO ON
6617	026752	104001			ERROR +1	;CPU ERROR
6618						;NO GO TO ERROR
6619	026754	020427	125250	2\$:	CMP R4,#125250	;IS R4 CORRECT
6620	026760	001401			BEQ 3\$;YES GO ON
6621	026762	104001			ERROR +1	;CPU ERROR
6622						;NO GO TO ERROR
6623	026764	020527	000000	3\$:	CMP R5,#0	;IS R5 CORRECT
6624	026770	001401			BEQ 4\$;YES GO ON

FIN127: NOP

TE130:

```

;
; TEST ASHC (ARITHMETIC SHIFT COMBINED)
CLR      @#177776          ;INIT PSW
MOV      #23,R1           ;SETUP R1
MOV      #52525,R5        ;SETUP R5
CLR      R4               ;SETUP R4
SCC      ;SET ALL CC BITS
ASHC     R1,R4            ; TEST INSTRUCTION
CMP      @#177776,#12     ;IS PS CORRECT
BEQ      1$               ;YES GO ON
ERROR    +1               ;CPU ERROR
;NO GO TO ERROR
1$:      CMP      R1,#23   ;IS R1 CORRECT
BEQ      2$               ;YES GO ON
ERROR    +1               ;CPU ERROR
;NO GO TO ERROR
2$:      CMP      R4,#125250 ;IS R4 CORRECT
BEQ      3$               ;YES GO ON
ERROR    +1               ;CPU ERROR
;NO GO TO ERROR
3$:      CMP      R5,#0    ;IS R5 CORRECT
BEQ      4$               ;YES GO ON
    
```

***** DOUBLE OPERAND TESTS *****

```

6625 026772 104001          ERROR +1          ;CPU ERROR
6626                                     ;NO GO TO ERROR
6627 026774 012703 052525  4$:  MOV    #52525,R3          ;SETUP R3
6628 027000 005002          CLR    R2              ;SETUP R2
6629 027002 012704 164731  MOV    #164731,R4       ;SETUP R4
6630 027006 000277          SCC              ;SET ALL CC BITS
6631 027010 073327 000023  ASHC   #23,R3          ; TEST INSTRUCTION
6632 027014 023727 177776 000012  CMP    #177776,#12     ;IS PS CORRECT
6633 027022 001401          BEQ    5$              ;YES GO ON
6634 027024 104001          ERROR +1          ;CPU ERROR
6635                                     ;NO GO TO ERROR
6636 027026 020227 000000  5$:  CMP    R2,#0          ;IS R2 CORRECT
6637 027032 001401          BEQ    6$              ;YES GO ON
6638 027034 104001          ERROR +1          ;CPU ERROR
6639                                     ;NO GO TO ERROR
6640 027036 020327 000000  6$:  CMP    R3,#0          ;IS R3 CORRECT
6641 027042 001401          BEQ    7$              ;YES GO ON
6642 027044 104001          ERROR +1          ;CPU ERROR
6643                                     ;NO GO TO ERROR
6644 027046 020427 164731  7$:  CMP    R4,#164731     ;IS R4 CORRECT
6645 027052 001401          BEQ    8$              ;YES GO ON
6646 027054 104001          ERROR +1          ;CPU ERROR
6647                                     ;NO GO TO ERROR
6648                                     ;
6649                                     ;
6650 027056 012701 027166  8$:  MOV    #TE130A,R1       ;SETUP POINTERS TO TABLES
6651                                     ;
6652 027062 010104          9$:  MOV    R1,R4          ;SAVE A COPY OF R1
6653 027064 016102 000002  MOV    2(R1),R2        ;SETUP R2
6654 027070 016103 000004  MOV    4(R1),R3        ;SETUP R3
6655 027074 000277          SCC              ;SET ALL CC BITS
6656 027076 073211          ASHC   (R1),R2        ; TEST INSTRUCTION
6657 027100 023761 177776 000006  CMP    #177776,6(R1)  ;IS PS CORRECT
6658 027106 001401          BEQ    10$             ;YES GO ON
6659 027110 104001          ERROR +1          ;CPU ERROR
6660                                     ;NO GO TO ERROR
6661 027112 026102 000010 10$:  CMP    10(R1),R2       ;IS R2 CORRECT
6662 027116 001401          BEQ    11$             ;YES GO ON
6663 027120 104001          ERROR +1          ;CPU ERROR
6664                                     ;NO GO TO ERROR
6665 027122 026103 000012 11$:  CMP    12(R1),R3       ;IS R3 CORRECT
6666 027126 001401          BEQ    12$             ;YES GO ON
6667 027130 104001          ERROR +1          ;CPU ERROR
6668                                     ;NO GO TO ERROR
6669 027132 020401          12$:  CMP    R4,R1          ;IS R1 CORRECT
6670 027134 001402          BEQ    13$             ;YES GO ON
6671 027136 104001          ERROR +1          ;CPU ERROR
6672                                     ;NO GO TO ERROR
6673 027140 010401          MOV    R4,R1
6674 027142 021114          13$:  CMP    (R1),(R4)       ;IS SOURCE CORRECT
6675 027144 001401          BEQ    14$             ;YES GO ON
6676 027146 104001          ERROR +1          ;CPU ERROR
6677                                     ;NO GO TO ERROR
6678                                     ;POSSIBLE SOURCE CODE CORRUPTION
6679 027150 062701 000014 14$:  ADD    #14,R1          ;GO TO NEXT TEST
6680 027154 020127 027576  CMP    R1,#FIN130     ;ARE WE DONE
6681 027160 001340          BNE    9$              ;NO GO TO 9$

```

***** DOUBLE OPERAND TESTS *****

```

6682
6683
6684 027162 000167 000410          JMP      FIN130
6685
6686          :
6687 027166 177700          TE130A: .WORD 177700          ;SOURCE
6688 027170 100125          .WORD 100125          ;DESTINATION WORD 1
6689 027172 177777          .WORD 177777          ;DESTINATION WORD 2
6690 027174 000010          .WORD 10              ;TEST PSW
6691 027176 100125          .WORD 100125          ;RESULT WORD 1
6692 027200 177777          .WORD 177777          ;RESULT WORD 2
6693
6694 027202 177777          .WORD 177777          ;SOURCE
6695 027204 000001          .WORD 1              ;DESTINATION WORD 1
6696 027206 000000          .WORD 0              ;DESTINATION WORD 2
6697 027210 000000          .WORD 0              ;TEST PSW
6698 027212 000000          .WORD 0              ;RESULT WORD 1
6699 027214 100000          .WORD 100000         ;RESULT WORD 2
6700
6701 027216 177701          .WORD 177701         ;SOURCE
6702 027220 047777          .WORD 47777          ;DESTINATION WORD 1
6703 027222 100000          .WORD 100000         ;DESTINATION WORD 2
6704 027224 000012          .WORD 12             ;TEST PSW
6705 027226 117777          .WORD 117777         ;RESULT WORD 1
6706 027230 000000          .WORD 0              ;RESULT WORD 2
6707
6708 027232 177706          .WORD 177706         ;SOURCE
6709 027234 004256          .WORD 4256           ;DESTINATION WORD 1
6710 027236 177700          .WORD 177700         ;DESTINATION WORD 2
6711 027240 000002          .WORD 2              ;TEST PSW
6712 027242 025677          .WORD 25677          ;RESULT WORD 1
6713 027244 170000          .WORD 170000         ;RESULT WORD 2
6714
6715 027246 177711          .WORD 177711         ;SOURCE
6716 027250 065700          .WORD 65700          ;DESTINATION WORD 1
6717 027252 000012          .WORD 12             ;DESTINATION WORD 2
6718 027254 000013          .WORD 13             ;TEST PSW
6719 027256 100000          .WORD 100000         ;RESULT WORD 1
6720 027260 012000          .WORD 12000          ;RESULT WORD 2
6721
6722 027262 177737          .WORD 177737         ;SOURCE
6723 027264 000000          .WORD 0              ;DESTINATION WORD 1
6724 027266 000001          .WORD 1              ;DESTINATION WORD 2
6725 027270 000004          .WORD 4              ;TEST PSW
6726 027272 000000          .WORD 0              ;RESULT WORD 1
6727 027274 000000          .WORD 0              ;RESULT WORD 2
6728
6729 027276 177736          .WORD 177736         ;SOURCE
6730 027300 000000          .WORD 0              ;DESTINATION WORD 1
6731 027302 000001          .WORD 1              ;DESTINATION WORD 2
6732 027304 000000          .WORD 0              ;TEST PSW
6733 027306 040000          .WORD 40000          ;RESULT WORD 1
6734 027310 000000          .WORD 0              ;RESULT WORD 2
6735
6736 027312 177740          .WORD 177740         ;SOURCE
6737 027314 100000          .WORD 100000         ;DESTINATION WORD 1
6738 027316 000000          .WORD 0              ;DESTINATION WORD 2

```


***** DOUBLE OPERAND TESTS *****

6739	027320	000011	.WORD	11	;TEST PSW
6740	027322	177777	.WORD	177777	;RESULT WORD 1
6741	027324	177777	.WORD	177777	;RESULT WORD 2
6742					
6743	027326	177725	.WORD	177725	;SOURCE
6744	027330	177777	.WORD	177777	;DESTINATION WORD 1
6745	027332	174000	.WORD	174000	;DESTINATION WORD 2
6746	027334	000007	.WORD	7	;TEST PSW
6747	027336	000000	.WORD	0	;RESULT WORD 1
6748	027340	000000	.WORD	0	;RESULT WORD 2
6749					
6750	027342	177724	.WORD	177724	;SOURCE
6751	027344	177777	.WORD	177777	;DESTINATION WORD 1
6752	027346	174000	.WORD	174000	;DESTINATION WORD 2
6753	027350	000011	.WORD	11	;TEST PSW
6754	027352	100000	.WORD	100000	;RESULT WORD 1
6755	027354	000000	.WORD	0	;RESULT WORD 2
6756					
6757	027356	177733	.WORD	177733	;SOURCE
6758	027360	177777	.WORD	177777	;DESTINATION WORD 1
6759	027362	157023	.WORD	157023	;DESTINATION WORD 2
6760	027364	000012	.WORD	12	;TEST PSW
6761	027366	114000	.WORD	114000	;RESULT WORD 1
6762	027370	000000	.WORD	0	;RESULT WORD 2
6763					
6764	027372	177727	.WORD	177727	;SOURCE
6765	027374	000000	.WORD	0	;DESTINATION WORD 1
6766	027376	177777	.WORD	177777	;DESTINATION WORD 2
6767	027400	000013	.WORD	13	;TEST PSW
6768	027402	177600	.WORD	177600	;RESULT WORD 1
6769	027404	000000	.WORD	0	;RESULT WORD 2
6770					
6771	027406	177717	.WORD	177717	;SOURCE
6772	027410	177777	.WORD	177777	;DESTINATION WORD 1
6773	027412	000001	.WORD	1	;DESTINATION WORD 2
6774	027414	000011	.WORD	11	;TEST PSW
6775	027416	100000	.WORD	100000	;RESULT WORD 1
6776	027420	100000	.WORD	100000	;RESULT WORD 2
6777					
6778	027422	177741	.WORD	177741	;SOURCE
6779	027424	100000	.WORD	100000	;DESTINATION WORD 1
6780	027426	000000	.WORD	0	;DESTINATION WORD 2
6781	027430	000010	.WORD	10	;TEST PSW
6782	027432	177777	.WORD	177777	;RESULT WORD 1
6783	027434	177777	.WORD	177777	;RESULT WORD 2
6784					
6785	027436	177742	.WORD	177742	;SOURCE
6786	027440	037777	.WORD	37777	;DESTINATION WORD 1
6787	027442	177777	.WORD	177777	;DESTINATION WORD 2
6788	027444	000005	.WORD	5	;TEST PSW
6789	027446	000000	.WORD	0	;RESULT WORD 1
6790	027450	000000	.WORD	0	;RESULT WORD 2
6791					
6792	027452	177742	.WORD	177742	;SOURCE
6793	027454	077777	.WORD	77777	;DESTINATION WORD 1
6794	027456	177777	.WORD	177777	;DESTINATION WORD 2
6795	027460	000001	.WORD	1	;TEST PSW

***** DOUBLE OPERAND TESTS *****

```

6796 027462 000000      .WORD 0      ;RESULT WORD 1
6797 027464 000001      .WORD 1      ;RESULT WORD 2
6798
6799 027466 177711      .WORD 177711 ;SOURCE
6800 027470 065600      .WORD 65600  ;DESTINATION WORD 1
6801 027472 000012      .WORD 12     ;DESTINATION WORD 2
6802 027474 000003      .WORD 3     ;TEST PSW
6803 027476 000000      .WORD 0     ;RESULT WORD 1
6804 027500 012000      .WORD 12000 ;RESULT WORD 2
6805
6806 027502 177740      .WORD 177740 ;SOURCE
6807 027504 077777      .WORD 77777  ;DESTINATION WORD 1
6808 027506 177777      .WORD 177777 ;DESTINATION WORD 2
6809 027510 000004      .WORD 4     ;TEST PSW
6810 027512 000000      .WORD 0     ;RESULT WORD 1
6811 027514 000000      .WORD 0     ;RESULT WORD 2
6812
6813 027516 177737      .WORD 177737 ;SOURCE
6814 027520 177777      .WORD 177777 ;DESTINATION WORD 1
6815 027522 177774      .WORD 177774 ;DESTINATION WORD 2
6816 027524 000011      .WORD 11    ;TEST PSW
6817 027526 177777      .WORD 177777 ;RESULT WORD 1
6818 027530 177777      .WORD 177777 ;RESULT WORD 2
6819
6820 027532 177747      .WORD 177747 ;SOURCE
6821 027534 100000      .WORD 100000 ;DESTINATION WORD 1
6822 027536 174000      .WORD 174000 ;DESTINATION WORD 2
6823 027540 000010      .WORD 10    ;TEST PSW
6824 027542 177777      .WORD 177777 ;RESULT WORD 1
6825 027544 177700      .WORD 177700 ;RESULT WORD 2
6826
6827 027546 177753      .WORD 177753 ;SOURCE
6828 027550 006324      .WORD 6324  ;DESTINATION WORD 1
6829 027552 071002      .WORD 71002 ;DESTINATION WORD 2
6830 027554 000001      .WORD 1     ;TEST PSW
6831 027556 000000      .WORD 0     ;RESULT WORD 1
6832 027560 000146      .WORD 146   ;RESULT WORD 2
6833
6834 027562 177765      .WORD 177765 ;SOURCE
6835 027564 102351      .WORD 102351 ;DESTINATION WORD 1
6836 027566 177231      .WORD 177231 ;DESTINATION WORD 2
6837 027570 000011      .WORD 11    ;TEST PSW
6838 027572 177760      .WORD 177760 ;RESULT WORD 1
6839 027574 116477      .WORD 116477 ;RESULT WORD 2
6840 027576
6843 027576
6844
6845
6846 027576 005006      ;
6847 027600 112667 153314  CLR R6 ;CLEAR SP
6848 027604 022706 000002  MOVB (R6)+,COUNT ;TRY AUTOINC ON R6
6849 027610 001401      CMP #2,R6 ;VERIFY AUTO INC BY 2
6850      BEQ SPAU1 ;BRANCH IF GOOD
6851 027612 104001      ;BAD AUTO-INC
6852 027614 005006      SPAU1: ERROR +1 ;CPU ERROR
6853 027616 112667 153276  CLR R6 ;CLEAR R6
6854 027622 112667 153272  MOVB (R6)+,COUNT ;DOUBLE BYTE AUTO-INC
        MOVB (R6)+,COUNT

```

***** DOUBLE OPERAND TESTS *****

```

6855 027626 022706 000004      CMP      #4,R6          ;VERIFY RESULT
6856 027632 001401             BEQ      SPAU2         ;BRANCH IF GOOD
6857 027634 104001             ERROR   +1           ;CPU ERROR
6858                               ;BAD DOUBLE AUTO-INC
6859 0276  012706 001000      SPAU2:  MOV      #STBOT,R6      ;LOAD R6
6860 0276  114667 153252      MOV      (R6),COUNT        ; TEST AUTO-DEC
6861 027646 022706 000776      CMP      #776,R6         ;VERIFY RESULT
6862 027652 001401             BEQ      SPAU3         ;BRANCH IF GOOD
6863 027654 104001             ERROR   +1           ;CPU ERROR
6864
6865 027656 012706 001000      SPAU3:  MOV      #STBOT,R6      ;LOAD R6
6866 027662 114667 153232      MOV      -(R6),COUNT      ; TEST AUTO DEC
6867 027666 114667 153226      MOV      -(R6),COUNT      ; TEST AUTO-DEC
6868 027672 022706 000774      CMP      #774,R6         ;VERIFY RESULT
6869 027676 001401             BEQ      SPAU4         ;BRANCH IF GOOD
6870 027700 104001             ERROR   +1           ;CPU ERROR
6871
6872 027702 005006             SPAU4:  CLR      R6          ; TEST AUTO-INC ON SOP
6873 027704 105726             TSTB    (R6)+          ; TEST AUTO-INC
6874 027706 020627 000002      CMP      R6,#2         ;
6875 027712 001401             BEQ      SPAU5         ;BRANCH IF GOOD
6876 027714 104001             ERROR   +1           ;CPU ERROR
6877
6878 027716 012706 001000      SPAU5:  MOV      #STBOT,R6      ;LOAD R6
6879 027722 105746             TSTB    -(R6)         ; TEST AUTO-DEC
6880 027724 022706 000776      CMP      #776,R6         ;VERIFY RESULT
6881 027730 001401             BEQ      SPAU6         ;BRANCH IF GOOD
6882 027732 104001             ERROR   +1           ;CPU ERROR
6883
6884 027734 012706 001000      SPAU6:  MOV      #STBOT,R6
6885
6887
6889 027740             ;
6890             MTRY:
6891             ;
6892 027740 005067 150022      ; VERIFY YELLOW ZONE TRAP ON AUTO DEC OF R6
6893 027744 012706 000150      CLR      CPREG         ;INIT CPU ERROR REGISTER
6894             MOV      #150,R6      ;LOAD R6 WITH A VALUE THAT WILL
6895 027750 016767 150030 153034      MOV      4,SLOC00      ;CAUSE A YELLOW STACK TRAP(IE. <400)
6896 027756 012767 030014 150020      MOV      #MTRYA,4      ;SAVE VECTOR
6897 027764 016701 150156             MOV      146,R1        ;SETUP THE STACK OVERFLOW TRAP POINTER
6898 027770 016702 150150             MOV      144,R2        ;SAVE VECTOR
6899 027774 016703 150142             MOV      142,R3        ;SAVE VECTOR
6900 030000 005067 150142             CLR      146           ;SAVE VECTOR
6901 030004 005046             CLR      -(R6)         ;JUST AS A PRECAUTION
6902 030006 012706 001000      CLR      -(R6)         ;CAUSE A STACK OVERFLOW TRAP
6903 030012 104001             MOV      #STBOT,R6      ;RESTORE R6 FOR ERROR CALL
6904             ERROR   +1           ;CPU ERROR
6905             ;OVERFLOW TRAP FAILED
6906 030014             MTRYA:
6907 030014 022767 000010 147744      CMP      #BIT03,CPREG    ;WAS CPU ERROR REG SET PROPERLY?
6908 030022 001003             BNE     1$             ;GO TO ERROR IF NOT
6909 030024 020627 000142      CMP      R6,#142        ;VERIFY CORRECT DECREMENT OF R6
6910 030030 001401             BEQ     MTRYB          ;BRANCH IF GOOD
6911 030032 104001             1$:   ERROR   +1           ;CPU ERROR
6912             ;R6 IMPROPERLY DECREMENTED
6913 030034             MTRYB:             ;OR CPU ERROR REGISTER NOT CORRECT

```

***** DOUBLE OPERAND TESTS *****

```

6914 030034 005067 147726 CLR CPREG ;CLEAR THE CPU ERROR REGISTER
6915 030040 016767 152746 147736 MOV SLOC00,4 ;RESTORE VECTOR
6916 030046 010167 150074 MOV R1,146 ;RESTORE VECTORS
6917 030052 010267 150066 MOV R2,144 ;
6918 030056 010367 150060 MOV R3,142 ;
6919 030062 012706 001000 MOV #STBOT,R6 ;
6920
6921
6923 ;
6925 030066 ; MTRYM:
6926
6927 ; TEST STACK OVERFLOW TRAPS IN VARIOUS MODES
6928 030066 005067 147674 CLR CPREG ;CLEAR CPU ERROR REGISTER
6929 030072 012706 000400 MOV #400,R6 ;SETUP OVERFLOW R6 DATA
6930 030076 016767 147702 152706 MOV 4,SLOC00 ;SAVE VECTOR
6931 030104 012767 030126 147672 MOV #TRYMA,4
6932 030112 005067 150260 CLR 376 ;JUST AS A PRECAUTION
6933 030116 005046 CLR -(R6) ;CAUSE OVERFLOW TRAP
6934 030120 012706 001000 MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
6935 030124 104001 ERROR +1 ;CPU ERROR
6936 ;NO OVERFLOW TRAP
6937 030126 TRYMA:
6938 030126 005067 147634 CLR CPREG ;CLEAR CPU ERROR REGISTER
6939 030132 012705 001000 MOV #1000,R5 ;SETUP R5 DATA
6940 030136 012706 000400 MOV #400,R6 ;SETUP OVERFLOW R6 DATA
6941 030142 012767 030160 147634 MOV #TRYMB,4
6942 030150 064645 ADD -(R6),-(R5) ;CAUSE OVERFLOW TRAP
6943 030152 012706 001000 MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
6944 030156 104001 ERROR +1 ;CPU ERROR
6945 ;NO OVERFLOW TRAP
6946 030160 TRYMB:
6947 030160 005067 147602 CLR CPREG ;CLEAR CPU ERROR REGISTER
6948 030164 012706 000150 MOV #150,R6 ;SETUP OVERFLOW R6 DATA
6949 030170 012767 030206 147606 MOV #TRYMC,4
6950 030176 044546 BIC -(R5),-(R6) ;CAUSE OVERFLOW TRAP
6951 030200 012706 001000 MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
6952 030204 104001 ERROR +1 ;CPU ERROR
6953 ;NO OVERFLOW TRAP
6954 030206 005067 147554 TRYMC: CLR CPREG ;CLEAR CPU ERROR REGISTER
6955 030212 016767 152574 147564 MOV SLOC00,4 ;RESTORE VECTOR
6956 030220 012706 001000 MOV #STBOT,R6
6957
6958
6960 ;
6962 030224 ; MILLO:
6963
6964 ; TEST STACK OVERFLOW ON ILLEGAL INST TRAP
6965 030224 005067 147536 CLR CPREG ;CLEAR CPU ERROR REGISTER
6966 030230 012706 000400 MOV #400,R6 ;SETUP FOR OVERFLOW TRAP
6967 030234 016767 147550 152550 MOV 10,SLOC00 ;SAVE VECTOR
6968 030242 012767 030270 147540 MOV #MILLOA,10 ;SETUP ILLEGAL TRAP VECTOR
6969 030250 016767 147530 152536 MOV 4,SLOC01 ;SAVE VECTOR
6970 030256 012767 030276 147520 MOV #MILLOB,4 ;SETUP OVERFLOW TRAP VECTOR
6971 030264 000077 77 ;UNUSED INSTRUCTION TRAP
6972 030266 000240 NOP
6973 030270 012706 001000 MILLOA: MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
6974 030274 104001 ERROR +1 ;CPU ERROR

```

***** DOUBLE OPERAND TESTS *****

```

6975
6976 030276
6977 030276 016767 152512 147500
6978 030304 016767 152502 147476
6979 030312 005067 147450
6980 030316 012706 001000
6981
6982
6983
6985
6987 030322
6988
6989
6990 030322 005067 147440
6991 030326 012706 000400
6992 030332 016767 147462 152452
6993 030340 012767 030366 147452
6994 030346 016767 147432 152440
6995 030354 012767 030374 147422
6996
6997 030362 000004
6998 030364 000240
6999 030366 012706 001000
7000 030372 104001
7001
7002 030374
7003 030374 005067 147366
7004 030400 012706 001000
7005 030404 016767 152404 147372
7006 030412 016767 152374 147400
7007
7009
7010
7012 030420
7013
7014
7015 030420 005067 147342
7016 030424 012706 000400
7017 030430 016767 147374 152354
7018 030436 012767 030464 147364
7019 030444 016767 147334 152342
7020 030452 012767 030472 147324
7021
7022 030460 104000
7023 030462 000240
7024 030464 012706 001000
7025 030470 104001
7026
7027 030472
7028 030472 016767 152314 147330
7029 030500 016767 152310 147276
7030 030506 005067 147254
7031 030512 012706 001000
7032
7035 030516
7036
7037

```

;UNUSED INSTRUCTION TRAP

```

MILLOB:
MOV SLOC01,4 ;RESTORE VECTOR
MOV SLOC00,10 ;RESTORE VECTOR
CLR CPEREG ;CLEAR CPU ERROR REGISTER
MOV #STBOT,R6 ;RESTORE R6

;
MIOTO:
; TEST STACK OVERFLOW ON IOT TRAP
CLR CPEREG ;CLEAR CPU ERROR REGISTER
MOV #400,R6 ;SETUP STACK FOR OVERFLOW
MOV 20,SLOC00 ;SAVE OLD IOT VECTOR
MOV #IOTOA,20 ;SETUP ERROR ACTION ON IOT
MOV 4,SLOC01 ;SAVE VECTOR
MOV #IOTOB,4 ;SETUP CORRECT TRAP VECTOR FOR
;OVERFLOW
; TEST INSTRUCTION
IOT
NOP
IOTOA: MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
ERROR +1 ;CPU ERROR
;FAILURE OF STACK OVERFLOW

IOTOB:
CLR CPEREG ;CLEAR CPU ERROR REGISTER
MOV #STBOT,R6
MOV SLOC01,4 ;RESTORE VECTOR
MOV SLOC00,20 ;RESTORE TRAP VECTOR

;
MEMTO:
; TEST STACK OVERFLOW ON EMT TRAP
CLR CPEREG ;CLEAR CPU ERROR REGISTER
MOV #400,R6 ;SETUP STACK FOR OVERFLOW
MOV 30,SLOC00 ;SAVE OLD EMT VECTOR
MOV #EMTOA,30 ;SETUP ERROR ACTION ON EMT
MOV 4,SLOC01 ;SAVE VECTOR
MOV #EMTOB,4 ;SETUP CORRECT TRAP VECTOR FOR
;OVERFLOW
; TEST INSTRUCTION
EMTOA: MOV #STBOT,R6 ;RESTORE R6 FOR ERROR CALL
ERROR +1 ;CPU ERROR
;FAILURE OF STACK OVERFLOW

EMTOB:
MOV SLOC00,30 ;RESTORE TRAP VECTOR
MOV SLOC01,4 ;RESTORE VECTOR
CLR CPEREG ;CLEAR CPU ERROR REGISTER
MOV #STBOT,R6

MTRPO:
; TEST STACK OVERFLOW ON TRAP

```

***** DOUBLE OPERAND TESTS *****

```

7038 030516 005067 147244          CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7039 030522 012706 000400          MOV      #400,R6        ;SETUP STACK FOR OVERFLOW
7040 030526 016767 147302 152256  MOV      34,SLOC00      ;SAVE OLD TRP VECTOR
7041 030534 012767 030562 147272  MOV      #TRPOA,34      ;SETUP ERROR ACTION ON TRP
7042 030542 016767 147236 152244  MOV      4,SLOC01       ;SAVE VECTOR
7043 030550 012767 030570 147226  MOV      #TRPOB,4       ;SETUP CORRECT TRAP VECTOR FOR
7044                                     ;OVERFLOW
7045 030556 104400                TRAP                                     ; TEST INSTRUCTION
7046 030560 000240                NOP                                     ;
7047 030562 012706 001000          TRPOA: MOV      #STBOT,R6          ;RESTORE R6 FOR ERROR CALL
7048 030566 104001                ERROR      +1          ;CPU ERROR
7049                                     ;FAILURE OF STACK OVERFLOW
7050 030570                TRPOB:
7051 030570 016767 152216 147236  MOV      SLOC00, 4      ;RESTORE TRAP VECTOR
7052 030576 016767 152212 147200  MOV      SLOC01,4      ;RESTORE VECTOR
7053 030604 005067 147156          CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7054 030610 012706 001000          MOV      #STBOT,R6
7055
7057
7058
7060 030614                ;MBPTO:
7061
7062                ;
7063 030614 005067 147146          CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7064 030620 012706 000400          MOV      #400,R6        ;SETUP STACK FOR OVERFLOW
7065 030624 016767 147164 152160  MOV      14,SLOC00      ;SAVE OLD BPT VECTOR
7066 030632 012767 030660 147154  MOV      #BPTOA,14      ;SETUP ERROR ACTION ON BPT
7067 030640 016767 147140 152146  MOV      4,SLOC01       ;SAVE VECTOR
7068 030646 012767 030666 147130  MOV      #BPTOB,4       ;SETUP CORRECT TRAP VECTOR FOR
7069                                     ;OVERFLOW
7070 030654 000003                BPT                                     ; TEST INSTRUCTION
7071 030656 000240                NOP                                     ;
7072 030660 012706 001000          BPTOA: MOV      #STBOT,R6          ;RESTORE R6 FOR ERROR CALL
7073 030664 104001                ERROR      +1          ;CPU ERROR
7074                                     ;FAILURE OF STACK OVERFLOW
7075 030666                BPTOB:
7076 030666 005067 147074          CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7077 030672 016767 152114 147114  MOV      SLOC00,14      ;RESTORE TRAP VECTOR
7078 030700 016767 152110 147076  MOV      SLOC01,4      ;RESTORE VECTOR
7079 030706 012706 001000          MOV      #STBOT,R6
7080
7082
7083
7085 030712                ;MILAO:
7086
7087                ;
7088 030712 005067 147050          CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7089 030716 012706 000400          MOV      #400,R6        ;SETUP STACK FOR OVERFLOW
7090 030722 016767 147062 152062  MOV      10,SLOC00      ;SAVE OLD ILLEGAL INST. VECTOR
7091 030730 012767 030760 147052  MOV      #ILAOA,10      ;SETUP ERROR ACTION ILLEGAL OPCODE
7092 030736 016767 147042 152050  MOV      4,SLOC01       ;SAVE VECTOR
7093 030744 012767 030766 147032  MOV      #ILBOB,4       ;SETUP CORRECT TRAP VECTOR FOR
7094                                     ;OVERFLOW
7095 030752 005001                CLR      R1
7096 030754 000101                JMP      R1
7097 030756 000240                NOP
7098 030760 012706 001000          ILAOA: MOV      #STBOT,R6          ;RESTORE R6 FOR ERROR CALL

```

***** DOUBLE OPERAND TESTS *****

```

7099 030764 104001          ERROR +1          ;CPU ERROR
7100                                     ;FAILURE OF STACK OVERFLOW
7101 030766                                     ILBOB:
7102 030766 016767 152022 147010          MOV      SLOC01,4          ;RESTORE VECTOR
7103 030774 016767 152012 147006          MOV      SLOC00,10        ;RESTORE TRAP VECTOR
7104 031002 005067 146760                                     CLR      CPEREG          ;CLEAR CPU ERROR REGISTER
7105 031006 012706 001000          MOV      @STBOT,R6
7106
7108
7109
7111 031012                                     ;
7112                                     ;
7113                                     ;
7114 031012 012706 000400          ;          TEST STACK OVERFLOW ON ILLEGAL JSR INST
7115 031016 016767 146766 151766          MOV      @400,R6          ;SETUP STACK FOR OVERFLOW
7116 031024 012767 031054 146756          MOV      10,SLOC00        ;SAVE OLD VECTOR
7117 031032 016767 146746 151754          MOV      @ILLBOA,10       ;SETUP ERROR ACTION ON ILL. OPCODE
7118 031040 012767 031062 146736          MOV      4,SLOC01         ;SAVE VECTOR
7119                                     MOV      @ILLBUB,4        ;SETUP CORRECT TRAP VECTOR FOR
7120 031046 005001                                     CLR      R1                ;OVERFLOW
7121 031050 004501                                     JSR      R5,R1              ; TEST INSTRUCTION
7122 031052 000240                                     NOP
7123 031054 012706 001000          ILLBOA: MOV      @STBOT,R6          ;RESTORE R6 FOR ERROR CALL
7124 031060 104001          ERROR +1          ;CPU ERROR
7125                                     ;FAILURE OF STACK OVERFLOW
7126 031062                                     ILLBOB:
7127 031062 016767 151726 146714          MOV      SLOC01,4          ;RESTORE VECTOR
7128 031070 016767 151716 146712          MOV      SLOC00,10        ;RESTORE TRAP VECTOR
7129 031076 012706 001000          MOV      @STBOT,R6
7130
7132
7133
7135 031102                                     ;
7136                                     ;
7137                                     ;
7138 031102 016767 146676 151702          ;          TEST FOR FALSE STACK OVERFLOW
7139 031110 012767 031156 146666          MOV      4,SLOC00         ;SAVE VECTOR
7140 031116 012706 001002          MOV      @MSTOE,4         ;ANTICIPATE OVERFLOW ERROR
7141 031122 005746          MOV      @1002,R6         ;SETUP LEGAL R6
7142 031124 012706 002002          TST      -(R6)            ;TRY TO CAUSE STACK OVERFLOW
7143 031130 005746          MOV      @2002,R6         ;SETUP LEGAL R6
7144 031132 012706 004002          TST      -(R6)            ;TRY TO CAUSE STACK OVERFLOW
7145 031136 005746          MOV      @4002,R6         ;SETUP LEGAL R6
7146 031140 012706 010002          TST      -(R6)            ;TRY TO CAUSE STACK OVERFLOW
7147 031144 005746          MOV      @10002,R6        ;SETUP LEGAL R6
7148 031146 012706 100402          TST      -(R6)            ;TRY TO CAUSE STACK OVERFLOW
7149 031152 005746          MOV      @100402,R6       ;SETUP LEGAL R6
7150 031154 000403          BR       MSTOEE           ;EXIT MODULE
7151 031156 012706 001000          MSTOE:  MOV      @STBOT,R6          ;RESTORE R6 FOR ERROR CALL
7152 031162 104001          ERROR +1          ;CPU ERROR
7153                                     ;STACK OVERFLOW ERROR
7154 031164 016767 151622 146612          MSTOEE: MOV      SLOC00,4          ;RESTORE VECTOR
7155 031172 012706 001000          MOV      @STBOT,R6
7156
7158
7159
7161 031176          ;
          ;
          MTT:

```

***** DOUBLE OPERAND TESTS *****

```

7162
7163
7164 031176 012706 001000 ; TEST T-BIT TRAPS
7165 031202 016767 146606 151602 MOV #STBOT,R6 ;SETUP STACK
7166 031210 012746 000020 MOV 14,SLOC00 ;SAVE OLD T-BIT VECTOR
7167 031214 012746 031232 MOV #20,-(R6) ;PUSH T-BIT
7168 031220 012767 031234 146566 MOV #MTTA,-(R6) ;SETUP ERROR TRAP VECTOR
7169 031226 000002 MOV #MTTB,14 ;SETUP NEW T-BIT VECTOR
7170 031230 104001 RTI ;CAUSE A T BIT SET IN PSW
7171 ERROR +1 ;CPU ERROR
7172 031232 104001 MTTA: ERROR +1 ;SHOULD NEVER BE EXECUTED
7173 ;CPU ERROR
7174 031234 022706 000774 MTTB: CMP #STBOT-4,R6 ;DIDNT TAKE CORRECT TRAP
7175 031240 001401 BEQ MTTD ;VERIFY SP DECIRMENT
7176 031242 104001 ERROR +1 ;BRANCH IF GOOD
7177 ;CPU ERROR
7178 031244 021627 031232 MTTD: CMP (R6),#MTTA ;BAD SP
7179 031250 001401 BEQ MTTE ;VERIFY PC SAVED ON STACK
7180 031252 104001 ERROR +1 ;BRANCH IF GOOD
7181 ;CPU ERROR
7182 031254 MTTE: ;INCORRECT PC ON STACK
7183 031254 016767 151532 146532 MOV SLOC00,14 ;RESTORE VECTOR 14
7184
7185 031262 012706 001000 MOV #STBOT,R6
7186
7188 ;
7190 031266 MTTT:
7191 ;
7192 ; TEST T-BIT TRAPS WITH RTT
7193 031266 012706 001000 MOV #STBOT,R6 ;SETUP STACK
7194 031272 016767 146516 151512 MOV 14,SLOC00 ;SAVE OLD T-BIT VECTOR
7195 031300 012746 000020 MOV #20,-(R6) ;PUSH T-BIT
7196 031304 012746 031322 MOV #MTTSA,-(R6) ;SETUP ERROR TRAP VECTOR
7197 031310 012767 031326 146476 MOV #MTTSB,14 ;SETUP NEW T-BIT VECTOR
7198 031316 000006 RTT ;CAUSE A T BIT SET IN PSW
7199 031320 104001 ERROR +1 ;CPU ERROR
7200 ;SHOULD NEVER BE EXECUTED
7201 031322 000240 MTTSA: NOP ;RTT WILL EXECUTE THIS INSTRUCTION
7202 ;WITH A T-BIT TRAP
7203 031324 104001 MTTSQ: ERROR +1 ;CPU ERROR
7204 ;DIDNT TAKE CORRECT TRAP
7205 031326 022706 000774 MTTSB: CMP #STBOT-4,R6 ;VERIFY SP DECIRMENT
7206 031332 001401 BEQ MTTSD ;BRANCH IF GOOD
7207 031334 104001 ERROR +1 ;CPU ERROR
7208 ;BAD SP
7209 031336 021627 031324 MTTSD: CMP (R6),#MTTSQ ;VERIFY PC SAVED ON STACK
7210 031342 001401 BEQ MTTSE ;BRANCH IF GOOD
7211 031344 104001 ERROR +1 ;CPU ERROR
7212 ;INCORRECT PC ON STACK
7213 031346 MTTSE:
7214 031346 016767 151440 146440 MOV SLOC00,14 ;RESTORE VECTOR 14
7215 031354 012706 001000 MOV #STBOT,R6
7216
7219 031360 MTTR:
7220 ;
7221 ; TEST OLD STATUS ON T-BIT TRAP
7222 031360 012706 001000 MOV #STBOT,R6 ;SETUP STACK

```


***** DOUBLE OPERAND TESTS *****

```

7223 031364 016767 146424 151420      MOV      14,SLOC00      ;SAVE OLD VECTOR
7224 031372 012746 000020                MOV      #20,-(R6)      ;PUSH T-BIT
7225 031376 012746 031424                MOV      #MTTRA,-(R6)  ;SETUP ERROR TRAP VECTOR
7226 031402 012767 031426 146404      MOV      #MTTRB,14     ;SETUP NEW T-BIT VECTOR
7227 031410 012767 000357 146360      MOV      #357,PS      ;SET PRIORITY AND COND C
7228 031416 000277                SCC
7229 031420 000002                RTI
7230 031422 104001                ERROR      +1          ;CPU ERROR
7231                                ;SHOULD NEVER EXECUTE
7232 031424 104001                MTTRA:  ERROR      +1          ;CPU ERROR
7233                                ;DIDNT TAKE CORRECT TRAP
7234 031426 026727 147344 000020      MTTRB:  CMP      STBOT-2,#20      ;VERIFY PSW ON STACK
7235 031434 001401                BEQ      MTTRC          ;BRANCH IF CORRECT STATUS
7236 031436 104001                ERROR      +1          ;CPU ERROR
7237                                ;BAD STATUS ON STACK
7238 031440 012706 001000                MTTRC:  MOV      #STBOT,R6      ;SETUP STACK
7239 031444 012746 000377                MOV      #377,-(R6)     ;PUSH T-BIT
7240 031450 012746 031476                MOV      #MTTRD,-(R6)  ;SETUP ERROR TRAP VECTOR
7241 031454 012767 031500 146332      MOV      #MTTRE,14     ;SETUP NEW T-BIT VECTOR
7242 031462 012767 000000 146306      MOV      #0,PS        ;CLEAR PRIORITY
7243 031470 000257                CCC          ;CLEAR CONDITION CODES
7244 031472 000002                RTI
7245 031474 104001                ERROR      +1          ;CPU ERROR
7246                                ;SHOULD NEVER EXECUTE
7247 031476 104001                MTTRD:  ERROR      +1          ;CPU ERROR
7248                                ;DIDNT TAKE CORRECT TRAP
7249 031500 026727 147272 000377      MTTRE:  CMP      STBOT-2,#377    ;VERIFY OLD PSW ON STACK
7250 031506 001401                BEQ      MTTRF         ;BRANCH IF GOOD
7251 031510 104001                ERROR      +1          ;CPU ERROR
7252                                ;OLD PSW INCORRECT
7253 031512                                ;MTTRF:
7254 031512 016767 151274 146274      MOV      SLOC00,14     ;RESTORE VECTOR
7255 031520 012706 001000                MOV      #STBOT,R6
7256
7258                                ;
7260 031524                                ;MRT:
7261
7262                                ;
7263 031524 012706 001000                ;      TEST RESERVED INST TRAP
7264 031530 016767 146254 151254      MOV      #STBOT,R6     ;SETUP STACK
7265 031536 012767 031550 146244      MOV      10,SLOC00     ;SAVE OLD VECTOR
7266 031544 000077                MOV      #MRTB,10     ;SETUP NEW RESERVED VECTOR
7267 031546 104001                77
7268                                ;CPU ERROR
7269 031550 022706 000774                MRTA:  ERROR      +1          ;DIDNT TAKE CORRECT TRAP
7270 031554 001401                MRTB:  CMP      #STBOT-4,R6     ;VERIFY SP DECRIMENT
7271 031556 104001                BEQ      MRTE          ;BRANCH IF GOOD
7272                                ;CPU ERROR
7273 031560 021627 031546                MRTE:  CMP      (R6),#MRTA     ;VERFY PROPER PC ON STACK
7274 031564 001401                BEQ      MRTF          ;BRANCH IF GOOD
7275 031566 104001                ERROR      +1          ;CPU ERROR
7276                                ;INCORRECT PC ON STACK
7277 031570                                ;MRTF:
7278 031570 016767 151216 146212      MOV      SLOC00,10     ;RESTORE TRAP VECTOR
7279 031576 012706 001000                MOV      #STBOT,R6
7280
7282                                ;

```

***** DOUBLE OPERAND TESTS *****

```

7283
7285 031602          ;
7286                ; MRT0:
7287                ;
7288 031602 012706 001000          ; TEST OLD STATUS ON RESERVED INST TRAP
7289 031606 016767 146176 151176  MOV   #STBOT,R6          ;SETUP STACK
7290 031614 012767 031634 146166  MOV   10,SLOC00        ;SAVE OLD VECTOR
7291 031622 005067 146150          MOV   #MRT0B,10       ;SETUP NEW VECTOR
7292 031626 000257          CLR   PS              ;CLEAR PRIORITY AND COND C
7293 031630 000077          CCC
7294 031632 104001          TRAP 77
7295                MRT0A: ERROR +1          ;CPU ERROR
7296 031634 026727 147136 000000  MRT0B: CMP   STBOT-2,#0      ;DIDNT TAKE CORRECT TRAP
7297 031642 001401          BEQ   MRT0C          ;VERIFY PSW ON STACK
7298 031644 104001          ERROR +1        ;BRANCH IF CORRECT STATUS
7299                ;CPU ERROR
7300 031646 012706 001000          MRT0C: MOV   #STBOT,R6          ;BAD STATUS ON STACK
7301 031652 012767 031674 146130  MOV   #MRT0E,10       ;SETUP STACK
7302 031660 012767 000357 146110  MOV   #357,PS         ;SET UP TRAP VECTOR
7303 031666 000277          SCC          ;SET PRIORITY
7304 031670 000077          TRAP 77          ;SET CONDITION CODES
7305                ;RESERVED INSTRUCTION
7306 031672 104001          MRT0D: ERROR +1          ;CPU ERROR
7307                ;DIDNT TAKE CORRECT TRAP
7308 031674 026727 147076 000357  MRT0E: CMP   STBOT-2,#357      ;VERIFY OLD PSW ON STACK
7309 031702 001401          BEQ   MRT0F          ;BRANCH IF GOOD
7310 031704 104001          ERROR +1        ;CPU ERROR
7311                ;OLD PSW INCORRECT
7312 031706          MRT0F:
7313 031706 016767 151100 146074  MOV   SLOC00,10       ;RESOTRE TRAP VECTOR
7314 031714 012706 001000          MOV   #STBOT,R6
7315
7318 031720          MTP:
7319
7320                ;
7321 031720 012706 001000          ; TEST TRAP INST
7322 031724 016767 146104 151060  MOV   #STBOT,R6          ;SETUP STACK
7323 031732 012767 031752 146074  MOV   34,SLOC00        ;SAVE OLD VECTOR
7324 031740 005067 146032          MOV   #MTPB,34       ;SETUP NEW TRAP VECTOR
7325 031744 000257          CLR   PS              ;CLEAR PRIORITY ABND COND C
7326 031746 104400          TRAP
7327 031750 104001          TRAP
7328                MTPR: ERROR +1          ;CPU ERROR
7329 031752 022706 000774          MTPB: CMP   #STBOT-4,R6      ;DIDNT TAKE CORRECT TRAP
7330 031756 001401          BEQ   MTPQ          ;VERIFY SP DECRIMENT
7331 031760 104001          ERROR +1        ;BRANCH IF GOOD
7332                ;CPU ERROR
7333 031762 021627 031750          MTPQ: CMP   (R6),#MTPR      ;BAD PC ON STACK
7334 031766 001401          BEQ   MTPF          ;VERFY PROPER PC ON STACK
7335 031770 104001          ERROR +1        ;BRANCH IF GOOD
7336                ;CPU ERROR
7337 031772          MTPF:
7338 031772 016767 151014 146034  MOV   SLOC00,34       ;INCORRECT PC ON STACK
7339 032000 012706 001000          MOV   #STBOT,R6          ;RESTORE VECTOR
7340
7342                ;
7343                ;

```

***** DOUBLE OPERAND TESTS *****

```

7345 032004          MTP0:
7346
7347          ;      TEST OLD STATUS SAVED ON TRAP
7348 032004 012706 001000          MOV      #STBOT,R6          ;SETUP STACK
7349 032010 016767 146020 150774          MOV      34,SLOC00          ;SAVE OLD VECTOR
7350 032016 012767 032036 146010          MOV      #MTP0B,34          ;SETUP NEW TRAP VECTOR
7351 032024 005067 145746          CLR      PS          ;CLEAR PRIORITY AND COND C
7352 032030 000257          CCC
7353 032032 104400          TRAP
7354 032034 104001          MTP0A: ERROR +1          ;CPU ERROR
7355          ;DIDNT TAKE CORRECT TRAP
7356 032036 026727 146734 000000          MTP0B: CMP      STBOT-2,#0          ;VERIFY PSW ON STACK
7357 032044 001401          BEQ      MTP0C          ;BRANCH IF CORRECT STATUS
7358 032046 104001          ERROR +1          ;CPU ERROR
7359          ;BAD STATUS ON STACK
7360 032050 012706 001000          MTP0C: MOV      #STBOT,R6          ;SETUP STACK
7361 032054 012767 032076 145752          MOV      #MTP0E,34          ;SET UP TRAP VECTOR
7362 032062 012767 000357 145706          MOV      #357,PS          ;SET PRIORITY
7363 032070 000277          SCC          ;SET CONDITION CODES
7364 032072 104400          TRAP          ;ISSUE TRAP
7365 032074 104001          MTP0D: ERROR +1          ;CPU ERROR
7366          ;DIDNT TAKE CORRECT TRAP
7367 032076 026727 146674 000357          MTP0E: CMP      STBOT-2,#357          ;VERIFY OLD PSW ON STACK
7368 032104 001401          BEQ      MTP0F          ;BRANCH IF GOOD
7369 032106 104001          ERROR +1          ;CPU ERROR
7370          ;OLD PSW INCORRECT
7371 032110          MTP0F:
7372 032110 016767 150676 145716          MOV      SLOC00,34          ;RESTORE TRAP VECTOR
7373 032116 012706 001000          MOV      #STBOT,R6
7374
7376          ;
7377          ;
7379 032122          MTPA:
7380
7381          ;      TEST ALL TRAP OPCODES - SELF MODIFYING
7382 032122 005003          CLR      R3          ;SETUP REGISTER TO INDICATE OPCODE
7383 032124 012706 001000          MOV      #STBOT,R6          ;SETUP STACK
7384 032130 016767 145700 150654          MOV      34,SLOC00          ;SAVE OLD VECTOR
7385 032136 016767 145642 150650          MOV      4,SLOC01          ;SAVE IN CASE OF HALT
7386 032144 012767 032174 145632          MOV      #MTPAH,4          ;SETUP HALT TRAP
7387 032152 012767 032176 145654          MOV      #MTPAA,34          ;SETUP NEW TRAP VECTOR
7388 032160 000167 000012          JMP      MTPAA          ;GO INTO LOOPING CODE
7389          ;
7390 032164 000000          MTPAL: HALT          ;SET TO A ZERO
7391 032166 104001          ERROR +1          ;CPU ERROR
7392          ;TRAP INSTRUCTION FAILED TO TRAP
7393          ;EXAMINE OPCCODE AT LOCATION MTPAL:
7394 032170 000167 000002          JMP      MTPAA          ;ATTEMPT TO GO ON
7395          ;
7396 032174 104001          MTPAH: ERROR +1          ;CPU ERROR
7397          ;ERROR, EITHER CANT MODIFY LOCATION MTPAL
7398          ;OR TRAP INSTRUCTION FAILED
7399 032176          MTPAA:
7400 032176 005203          INC      R3          ;GET NEXT OPCODE
7401
7402 032200 012706 001000          MOV      #STBOT,R6          ;RESTORE STACK
7403 032204 020327 000400          JMP      R3,#400          ;SEE IF LAST OPCODE

```

***** DOUBLE OPERAND TESTS *****

```

7404 032210 001406          BEQ   MTPAE          ;BRANCH IF DONE
7405 032212 012767 104400 177744  MOV   #104400,MTPAL ;TRAP OPCODE INTO LOCATION
7406 032220 060367 177740          ADD   R3,MTPAL      ;FORM TEST OPCODE
7407 032224 000757          BR    MTPAL        ;EXECUTE TEST
7408 032226          MTPAE:
7409
7410 032226 016767 150560 145600  MOV   SLOC00,34
7411 032234 016767 150554 145542  MOV   SLOC01,4      ;RESTORE VECTORS
7412
7413 032242 012706 001000          MOV   #STBOT,R6
7414
7416          ;
7417          ;
7419 032246          MIOT:
7420
7421          ; TEST IOT TRAP
7422 032246 012706 001000          MOV   #STBOT,R6      ;SETUP STACK
7423 032252 016767 145542 150532  MOV   20,SLOC00     ;SAVE OLD VECTOR
7424 032260 012767 032272 145532  MOV   #MIOTB,20     ;SETUP NEW IOT VECTOR
7425 032266 000004
7426 032270 104001          MIOTA: ERROR   +1      ;CPU ERROR
7427          ;DIDNT TAKE CORRECT TRAP
7428 032272 022706 000774          MIOTB: CMP   #STBOT-4,R6 ;VERIFY SP DECRIMENT
7429 032276 001401          BEQ   MIOTD          ;BRANCH IF GOOD
7430 032300 104001          ERROR   +1      ;CPU ERROR
7431          ;BAD PC ON STACK
7432 032302 021627 032270          MIOTD: CMP   (R6),#MIOTA ;VERFY PROPER PC ON STACK
7433 032306 001401          BEQ   MIOTF          ;BRANCH IF GOOD
7434 032310 104001          ERROR   +1      ;CPU ERROR
7435          ;INCORRECT PC ON STACK
7436 032312 016767 150474 145500  MIOTF: MOV   SLOC00,20 ;RESTORE VECTOR
7437 032320 012706 001000          MOV   #STBOT,R6
7438
7441 032324          MITO:
7442
7443          ; TEST OLD STATUS ON IOT TRAP
7444 032324 012706 001000          MOV   #STBOT,R6      ;SETUP STACK
7445 032330 016767 145464 150454  MOV   20,SLOC00     ;SAVE OLD VECTOR
7446 032336 012767 032356 145454  MOV   #MITOB,20     ;SETUP NEW IOT VECTOR
7447 032344 005067 145426          CLR   PS            ;CLEAR PRIORITY AND COND C
7448 032350 000257
7449 032352 000004
7450 032354 104001          MITOA: ERROR   +1      ;CPU ERROR
7451          ;DIDNT TAKE CORRECT TRAP
7452 032356 026727 146414 000000  MITOB: CMP   STBOT-2,#0 ;VERIFY PSW ON STACK
7453 032364 001401          BEQ   MITOC          ;BRANCH IF CORRECT STATUS
7454 032366 104001          ERROR   +1      ;CPU ERROR
7455          ;BAD STATUS ON STACK
7456 032370 012706 001000          MITOC: MOV   #STBOT,R6 ;SETUP STACK
7457 032374 012767 032416 145416  MOV   #MITOE,20     ;SET UP TRAP VECTOR
7458 032402 012767 000357 145366  MOV   #357,PS       ;SET PRIORITY
7459 032410 000277          SCC          ;SET CONDITION CODES
7460 032412 000004
7461 032414 104001          MITOD: ERROR   +1      ;CPU ERROR
7462          ;DIDNT TAKE CORRECT TRAP
7463 032416 026727 146354 000357  MITOE: CMP   STBOT-2,#357 ;VERIFY OLD PSW ON STACK
7464 032424 001401          BEQ   MITOF          ;BRANCH IF GOOD

```

***** DOUBLE OPERAND TESTS *****

```

7465 032426 104001          ERROR +1          ;CPU ERROR
7466                                     ;OLD PSW INCORRECT
7467 032430          MITOF:
7468 032430 016767 150356 145362      MOV     SLOC00,20          ;RESTORE VECTOR
7469 032436 012706 001000          MOV     @STBOT,R6
7470
7472
7474 032442          ;
7475          MET:
7476          ; TEST EMULATOR TRAP INSTRUCTION (EMT)
7477 032442 012706 001000      MOV     @STBOT,R6          ;SETUP STACK
7478 032446 016767 145356 150336      MOV     30,SLOC00        ;SAVE OLD VECTOR
7479 032454 012767 032510 145346      MOV     @METB,30        ;SETUP NEW EMT VECTOR
7480 032462 016767 145346 150324      MOV     34,SLOC01        ;SAVE TRAP VECTOR
7481 032470 012767 140776 145336      MOV     @#ERROR,34      ;SET UP TO HANDLE EMT ERROR
7482 032476 104000          EMT
7483 032500 104400          META: TRAP          ;TRAP ON ERROR
7484 032502 001057          .WORD 559.
7485 032504 000001          .WORD 1          ;CPUERR
7486 032506 000001          .WORD 1          ;ERRTN
7487
7488 032510 022706 000774          METB:  CMP     @STBOT-4,R6      ;DIDNT TAKE CORRECT TRAP
7489 032514 001401          BEQ     METD          ;VERIFY SP DECRIMENT
7490 032516 104001          ERROR +1          ;BRANCH IF GOOD
7491
7492 032520 021627 032500          METD:  CMP     (R6),@META      ;CPU ERROR
7493 032524 001401          BEQ     METF          ;BAD PC ON STACK
7494 032526 104001          ERROR +1          ;VERIFY PROPER PC ON STACK
7495
7496 032530 016767 150260 145276          METF:  MOV     SLOC01,34      ;BRANCH IF GOOD
7497 032536 016767 150250 145264          MOV     SLOC00,30      ;CPU ERROR
7498 032544 012706 001000          MOV     @STBOT,R6      ;INCORRECT PC ON STACK
7499
7501          ;
7503 032550          ;
7504          ;
7505          ; TEST OLD STATUS ON EMT TRAP
7506 032550 012706 001000      MOV     @STBOT,R6          ;SETUP STACK
7507 032554 016767 145250 150230      MOV     30,SLOC00        ;SAVE OLD VECTOR
7508 032562 012767 032624 145240      MOV     @METOB,30        ;SETUP NEW EMT VECTOR
7509 032570 016767 145240 150216      MOV     34,SLOC01        ;SAVE TRAP VECTOR
7510 032576 012767 140776 145230      MOV     @#ERROR,34      ;SET UP TRAP VECTOR
7511 032604 005067 145166          CLR     PS          ;CLEAR PRIORITY AND COND C
7512 032610 000257          CCC
7513 032612 104000          EMT
7514 032614 104400          METOA: TRAP
7515 032616 001062          .WORD 562
7516 032620 000001          .WORD 1          ;CPUERR
7517 032622 000001          .WORD 1          ;ERRTN
7518
7519 032624 026727 146146 000000          METOB: CMP     STBOT-2,@0      ;DIDNT TAKE CORRECT TRAP
7520 032632 001401          BEQ     METOC          ;VERIFY PSW ON STACK
7521 032634 104001          ERROR +1          ;BRANCH IF CORRECT STATUS
7522
7523 032636 012706 001000          METOC: MOV     @STBOT,R6          ;CPU ERROR
7524 032642 012767 032672 145160          MOV     @METOE,30      ;BAD STATUS ON STACK
7525 032650 012767 000357 145120          MOV     @357,PS        ;SETUP STACK
                                     ;SET UP TRAP VECTOR
                                     ;SET PRIORITY

```

***** DOUBLE OPERAND TESTS *****

```

7526 032656 000277          SCC          ;SET CONDITION CODES
7527 032660 104000          EMT
7528 032662 104400          METHOD: TRAP
7529 032664 001064          .WORD 564.
7530 032666 000001          .WORD 1          ;CPUERR
7531 032670 000001          .WORD 1          ;FRRTN
7532
7533 032672 026727 146100 000357 METOE: CMP STBOT-2,#357 ;DIDNT TAKE CORRECT TRAP
7534 032700 001401          BEQ METOF          ;VERIFY OLD PSW ON STACK
7535 032702 104001          ERROR +1          ;BRANCH IF GOOD
7536
7537 032704          METOF:
7538 032704 016767 150104 145122 MOV SLOC01,34      ;RESTORE VECTOR
7539 032712 016767 150074 145110 MOV SLOC00,30      ;RESTORE VECTOR
7540 032720 012706 001000 MOV #STBOT,R6
7541
7543
7544
7546 032724          ;
7547          ;
7548          ;
7549 032724 012706 001000          ; TEST BPT TRAP
7550 032730 016767 145060 150054 MOV #STBOT,R6      ;SETUP STACK
7551 032736 012767 032750 145050 MOV 14,SLOC00      ;SAVE OLD VECTOR
7552 032744 000003          MOV #MBTB,14      ;SETUP NEW BPT VECTOR
7553 032746 104001          BPT
7554          MBTA: ERROR +1          ;CPU ERROR
7555 032750 022706 000774          ;DIDNT TAKE CORRECT TRAP
7556 032754 001401          MBTB: CMP #STBOT-4,R6 ;VERIFY SP DECRIMENT
7557 032756 104001          BEQ MBTD          ;BRANCH IF GOOD
7558          ERROR +1          ;CPU ERROR
7559 032760 021627 032746          ;BAD PC ON STACK
7560 032764 001401          MBTD: CMP (R6),#MBTA ;VERFY PROPER PC ON STACK
7561 032766 104001          BEQ MBTF          ;BRANCH IF GOOD
7562          ERROR +1          ;CPU ERROR
7563 032770 016767 150016 145016 MBTF: MOV SLOC00,14 ;INCORRECT PC ON STACK
7564 032776 012706 001000          MOV #STBOT,R6      ;RESTORE VECTOR
7565
7567
7568
7570 033002          ;
7571          ;
7572          ;
7573 033002 012706 001000          ; TEST OLD STATUS ON BPT TRAP
7574 033006 016767 145002 147776 MOV #STBOT,R6      ;SETUP STACK
7575 033014 012767 033034 144772 MOV 14,SLOC00      ;SAVE OLD VECTOR
7576 033022 005067 144750          MOV #MBTOB,14      ;SETUP NEW BPT VECTOR
7577 033026 000257          CLR PS          ;CLEAR PRIORITY AND COND C
7578 033030 000003          CCC
7579 033032 104001          BPT
7580          MBTOA: ERROR +1          ;CPU ERROR
7581 033034 026727 145736 000000 MBTOB: CMP STBOT 2,#0 ;DIDNT TAKE CORRECT TRAP
7582 033042 001401          BEQ MBTOC          ;VERIFY PSW ON STACK
7583 033044 104001          ERROR +1          ;BRANCH IF CORRECT STATUS
7584          ;CPU ERROR
7585 033046 012706 001000          ;BAD STATUS ON STACK
7586 033052 012767 033074 144734 MBTOC: MOV #STBOT,R6 ;SETUP STACK
          MOV #MBTOE,14 ;SET UP TRAP VECTOR

```

***** DOUBLE OPERAND TESTS *****

```

7587 037 00 012767 000357 144710      MOV      #357,PS          ;SET PRIORITY
7588 033066 000277                      SCC                      ;SET CONDITION CODES
7589 033070 000003                      BPT
7590 033072 104001                      MBTOD:  ERROR      +1      ;CPU ERROR
7591                                         ;DIDNT TAKE CORRECT TRAP
7592 033074 026727 145676 000357 MBTOE:  CMP      STBOT-2,#357      ;VERIFY OLD PSW ON STACK
7593 033102 001401                      BEQ      MBTOF          ;BRANCH IF GOOD
7594 033104 104001                      ERROR      +1          ;CPU ERROR
7595                                         ;OLD PSW INCORRECT
7596 033106                                MBTOF:
7597 033106 016767 147700 144700      MOV      SLOC00,14      ;RESTORE VECTOR
7598 033114 012706 001000              MOV      #STBOT,R6
7599
7601                                     ;
7602                                     ;
7604 033120                                MIL:
7605
7606                                     ;      TEST ILLEGAL JUMP INSTRUCTION TRAP
7607 033120 012706 001000              MOV      #STBOT,R6      ;SETUP STACK
7608 033124 016767 144660 147660      MOV      10,SLOC00      ;SAVE OLD VECTOR
7609 033132 012767 033146 144650      MOV      #MILB,10      ;SETUP NEW ILLEGAL VECTOR
7610 033140 005001                      CLR      R1
7611 033142 000101                      JMP      R1
7612 033144 104001                      MILA:  ERROR      +1      ;**TEST INSTRUCTIO
7613                                         ;CPU ERROR
7614 033146 022706 000774                      MILB:  CMP      #STBOT-4,R6      ;DIDNT TAKE CORRECT TRAP
7615 033152 001401                      BEQ      MILD          ;VERIFY SP DECRIMENT
7616 033154 104001                      ERROR      +1          ;BRANCH IF GOOD
7617                                         ;CPU ERROR
7618 033156 021627 033144                      MILD:  CMP      (R6),#MILA      ;BAD PC ON STACK
7619 033162 001401                      BEQ      MILF          ;VERIFY PROPER PC ON STACK
7620 033164 104001                      ERROR      +1          ;BRANCH IF GOOD
7621                                         ;CPU ERROR
7622 033166 016767 147620 144614 MILF:  MOV      SLOC00,10      ;INCORRECT PC ON STACK
7623 033174 012706 001000              MOV      #STBOT,R6      ;RESTORE VECTOR
7624
7627 033200                                MILO:
7628
7629                                     ;      TEST OLD STATUS ON ILLEGAL JUMP TRAP
7630 033200 012706 001000              MOV      #STBOT,R6      ;SETUP STACK
7631 033204 016767 144600 147600      MOV      10,SLOC00      ;SAVE OLD VECTOR
7632 033212 012767 033234 144570      MOV      #MILOB,10      ;SETUP NEW ILLEGAL VECTOR
7633 033220 005067 144552                      CLR      PS            ;CLEAR PRIORITY AND COND C
7634 033224 000257                      CCC
7635 033226 005001                      CLR      R1
7636 033230 000101                      JMP      R1
7637 033232 104001                      MILOA:  ERROR      +1      ;CPU ERROR
7638                                         ;DIDNT TAKE CORRECT TRAP
7639 033234 026727 145536 000004 MILOB:  CMP      STBOT-2,#4      ;VERIFY PSW ON STACK
7640 033242 001401                      BEQ      MILOC        ;BRANCH IF CORRECT STATUS
7641 033244 104001                      ERROR      +1          ;CPU ERROR
7642                                         ;BAD STATUS ON STACK
7643 033246 012706 001000              MILOC:  MOV      #STBOT,R6      ;SETUP STACK
7644 033252 012767 033274 144530      MOV      #MILOE,10      ;SET UP TRAP VECTOR
7645 033260 012767 000357 144510      MOV      #357,PS        ;SET PRIORITY
7646 033266 000277                      SCC                      ;SET CONDITION CODES
7647 033270 000101                      JMP      R1

```

***** DOUBLE OPERAND TESTS *****

```

7648 033272 104001          MILOD:  ERROR  +1          ;CPU ERROR
7649                                ;DIDNT TAKE CORRECT TRAP
7650 033274 026727 145476 000357 MILOE:  CMP    STBOT-2,#357      ;VERIFY OLD PSW ON STACK
7651 033302 001401          BEQ    MILOF          ;BRANCH IF GOOD
7652 033304 104001          ERROR  +1          ;CPU ERROR
7653                                ;OLD PSW INCORRECT
7654 033306                MILOF:
7655 033306 016767 147500 144474      MOV    SLOC00,10      ;RESTORE VECTOR
7656 033314 012706 001000          MOV    #STBOT,R6
7657
7659
7660
7662 033320                MIALL:
7663
7664                ;      TEST ILLEGAL JSR INSTRUCTION TRAP
7665 033320 012706 001000          MOV    #STBOT,R6      ;SETUP STACK
7666 033324 016767 144460 147460      MOV    10,SLOC00      ;SAVE OLD VECTOR
7667 033332 012767 033346 144450      MOV    #MIALLB,10     ;SETUP NEW ILLEGAL VECTOR
7668 033340 005003          CLR    R3
7669 033342 004303          JSR   R3,R3
7670 033344 104001          MIALLA: ERROR  +1          ;CPU ERROR
7671                                ;DIDNT TAKE CORRECT TRAP
7672 033346 022706 000774          MIALLB: CMP    #STBOT-4,R6      ;VERIFY SP DECRIMENT
7673 033352 001401          BEQ    MIALLD          ;BRANCH IF GOOD
7674 033354 104001          ERROR  +1          ;CPU ERROR
7675                                ;BAD PC ON STACK
7676 033356 021627 033344          MIALLD: CMP    (R6),#MIALLA      ;VERFY PROPER PC ON STACK
7677 033362 001401          BEQ    MIALLF          ;BRANCH IF GOOD
7678 033364 104001          ERROR  +1          ;CPU ERROR
7679                                ;INCORRECT PC ON STACK
7680 033366 016767 147420 144414      MIALLF: MOV    SLOC00,10      ;RESTORE VECTOR
7681 033374 012706 001000          MOV    #STBOT,R6
7682
7684
7685
7687 033400                MJSI:
7688
7689                ;      TEST OLD STATUS ON ILLEGAL JSR TRAP
7690 033400 012706 001000          MOV    #STBOT,R6      ;SETUP STACK
7691 033404 016767 144400 147400      MOV    10,SLOC00      ;SAVE OLD VECTOR
7692 033412 012767 033434 144370      MOV    #MJSIB,10     ;SETUP NEW VECTOR
7693 033420 005067 144352          CLR    PS            ;CLEAR PRIORITY AND COND C
7694 033424 000257          CCC
7695 033426 005003          CLR    R3
7696 033430 004303          JSR   R3,R3
7697 033432 104001          MJSIA: ERROR  +1          ;CPU ERROR
7698                                ;DIDNT TAKE CORRECT TRAP
7699 033434 026727 145336 000004      MJSIB: CMP    STBOT-2,#4      ;VERIFY PSW ON STACK
7700 033442 001401          BEQ    MJSIC          ;BRANCH IF CORRECT STATUS
7701 033444 104001          ERROR  +1          ;CPU ERROR
7702                                ;BAD STATUS ON STACK
7703 033446 012706 001000          MJSIC: MOV    #STBOT,R6      ;SETUP STACK
7704 033452 012767 033474 144330      MOV    #MJSIE,10     ;SET UP TRAP VECTOR
7705 033460 012767 000357 144310      MOV    #357,PS       ;SET PRIORITY
7706 033466 000277          SCC          ;SET CONDITION CODES
7707 033470 004303          JSR   R3,R3
7708 033472 104001          MJSID: ERROR  +1          ;CPU ERROR

```


***** DOUBLE OPERAND TESTS *****

```

7709
7710 033474 026727 145276 000357 MJSIE: CMP STBOT-2,#357 ;DIDNT TAKE CORRECT TRAP
7711 033502 001401 BEQ MJSIF ;VERIFY OLD PSW ON STACK
7712 033504 104001 ERROR +1 ;BRANCH IF GOOD
7713 ;CPU ERROR
7714 033506 MJSIF: ;OLD PSW INCORRECT
7715 033506 016767 147300 144274 MOV SLOC00,10 ;RESTORE VECTOR
7716 033514 012706 001000 MOV #STBOT,R6
7717
7719 ;
7720 ;
7722 033520 IOXXX:
7723 ; I/O TIME OUT TEST
7724 033520 005067 144242 CLR CPEREG ;CLEAR CPU ERROR REGISTER
7725 033524 016767 144254 147260 MOV 4,SLOC00 ;SAVE VECTOR
7726 033532 012767 033554 144244 MOV #2#,4 ;SET UP VECTOR TO HANDLE NXM
7727 033540 012767 030000 144230 MOV #30000,PS ;INIT THE PSW TO A KNOWN STATE
7728 033546 005737 177700 TST #177700 ;TRY TO ACCESS HARDWARE ADDRESS
7729 ;FOR GENERAL PURPOSE REG 0. THIS
7730 ;IS NOT IMPLEMENTED ON KDJ11
7731 ;SHOULD CAUSE TIME OUT.
7732 033552 104001 1#: ERROR +1 ;CPU ERROR
7733 033554 022767 000020 144204 2#: CMP #BIT04,CPEREG ;IS CPU ERROR REGISTER CORRECT?
7734 033562 001401 BEQ 3#
7735 033564 104001 ERROR +1 ;CPU ERROR
7736 033566 022627 033552 3#: CMP (SP)+,#1# ;CHECK THAT STACK CONTAINS CORRECT ADDR.
7737 033572 001401 BEQ 4#
7738 033574 104001 ERROR +1 ;CPU ERROR
7739 033576 022627 030000 4#: CMP (SP)+,#30000 ;IS THE PSW OK?
7740 033602 001401 BEQ 5#
7741 033604 104001 ERROR +1 ;CPU ERROR
7742 033606 005067 144154 5#: CLR CPEREG ;CLEAR THE CPU ERROR REGISTER
7743 033612 016767 147174 144164 MOV SLOC00,4 ;RESTORE VECTOR
7744
7747 033620 ODDXX:
7756 ; ODD ADDRESS/ILLEGAL INST FETCH TRAP TEST
7757 ;*****
7758 ;THIS PROGRAM GENERATES AN ODD ADDRESS IN THE PC. THE KDJ11 SHOULD
;TRAP THROUGH ADDR 4
7759 033620 005067 144142 CLR CPEREG ;INIT THE CPU ERROR REG
7760 033624 016767 144154 147160 MOV 4,SLOC00 ;SAVE VECTOR
7761 033632 012767 033666 144144 MOV #2#,4 ;SET UP VECTOR TO HANDLE ODD ADDR TRAP
7762 033640 016767 144142 147146 MOV 6,SLOC01 ;SAVE VECTOR
7763 033646 005067 144134 CLR 6 ;INIT VECTOR
7764 033652 012746 030000 MOV #30000,-(SP) ;PUSH A KNOWN PSW ON THE STACK
7765 033656 012746 033665 MOV #1#+1,-(SP) ;PUSH AN ODD NUMBER ON THE STACK
7766 033662 000002 RTI ;POP ODD ADDRESS OFF STACK INTO PC
7767 ;SHOULD TRAP HERE
7768 033664 104001 1#: ERROR +1 ;CPU ERROR
7769 033666 022767 000100 144072 2#: CMP #BIT06,CPEREG ;IS CPU ERROR REGISTER CORRECT?
7770 033674 001401 BEQ 3#
7771 033676 104001 ERROR +1 ;CPU ERROR
7772 033700 022726 033665 3#: CMP #1#+1,(SP)+ ;IS STACK CONTENTS CORRECT?
7773 033704 001401 BEQ 4#
7774 033706 104001 ERROR +1 ;CPU ERROR
7775 033710 022726 030000 4#: CMP #30000,(SP)+ ;IS PSW CORRECT ON STACK?

```

***** DOUBLE OPERAND TESTS *****

```

7776 033714 001401          BEQ      5#
7777 033716 104001          ERROR    +1          ;CPU ERROR
7778 033720 005067 144042  5# :    CLR      CPEREG          ;CLEAR CPU ERROR REG
7779
7780          ;NOW WE'LL TRY TO FETCH AN INSTRUCTION FROM AN INTERNAL REGISTER.
7781          ;THIS SHOULD CAUSE A TRAP TO ADDR 4 AND SET BIT 6 IN THE CPU
7782          ;ERROR REGISTER.
7783          ;
7784 033724 012767 033752 144052  MOV      #7#,4          ;LOAD VECTOR WITH TRAP HANDLER ADDR
7785 033732 012767 030340 144046  MOV      #30340,6       ;LOAD VEC WITH PSW VALUE ON TRAP
7786 033740 005067 144032          CLR      PS             ;CLEAR THE PSW
7787 033744 000167 144026          JMP      PS             ;*****TEST INSTRUCTION*****
7788          ;TRY INSTRUCTION FETCH FROM INTERNAL
7789          ;REGISTER-- SHOULD TRAP VIA ADDR 4
7790 033750 104001          6# :    ERROR    +1          ;CPU ERROR
7791 033752 016701 144020          7# :    MOV      PS,R1       ;SAVE CONTENTS OF PSW IN R1
7792 033756 022767 000100 144002  CMP      #BIT06,CPEREG  ;IS CPU ERROR REGISTER CORRECT?
7793 033764 001401          BEQ      8#
7794 033766 104001          ERROR    +1          ;CPU ERROR
7795 033770 022726 177776          8# :    CMP      #PS,(SP)+   ;IS STACK CONTENTS CORRECT?
7796 033774 001401          BEQ      9#
7797 033776 104001          ERROR    +1          ;CPU ERROR
7798 034000 022726 000000          9# :    CMP      #0,(SP)+   ;IS STACK CONTENTS CORRECT?
7799 034004 001401          BEQ     10#
7800 034006 104001          ERROR    +1          ;CPU ERROR
7801 034010 022701 000340          10# :   CMP      #340,R1     ;HAS PSW LOADED PROPERLY ON TRAP?
7802 034014 001401          BEQ     11#
7803 034016 104001          ERROR    +1          ;CPU ERROR
7804 034020 005067 143742          11# :   CLR      CPEREG     ;CLEAR CPU ERROR REGISTER
7805 034024 016767 146762 143752  MOV      SLOC00,4       ;RESTORE VECTOR
7806 034032 016767 146756 143746  MOV      SLOC01,6       ;
7807
7810 034040          RXXX:
7811
7812          ; RED ZONE TRAP TEST
7813 034040 013767 000004 146744  MOV      #4,SLOC00     ;SAVE VECTOR
7814 034046 012737 034076 000004  MOV      #2#,R6        ;SET UP VECTOR
7815 034054 012706 000777          MOV      #777,R6      ;SET UP THE STACK WITH ODD ADDRESS.
7816 034060 005067 143702          CLR      CPEREG     ;CLEAR THE CPU ERROR REGISTER
7817 034064 005067 143706          CLR      PSW        ;CLEAR THE PSW
7818 034070 005737 177700          TST      #177700     ;ACCESS NON-EXISTANT I/O ADDRESS
7819 034074 104001          1# :    ERROR    +1          ;CPU ERROR
7820 034076 022706 000000          2# :    CMP      #0,SP     ;IS R6 CORRECT?
7821 034102 001401          BEQ      3#          ;BRANCH IF YES
7822 034104 104001          ERROR    +1          ;CPU ERROR
7823 034106 022726 034074          3# :    CMP      #1#,(SP)+  ;IS DATA AT ADDR 0 CORRECT?
7824 034112 001401          BEQ      4#          ;BRANCH IF YES
7825 034114 104001          ERROR    +1          ;CPU ERROR
7826 034116 022716 000000          4# :    CMP      #0,(SP)   ;IS PSW DATA IN ADDR 2 CORRECT?
7827 034122 001401          BEQ      5#          ;BRANCH IF YES
7828 034124 104001          ERROR    +1          ;CPU ERROR
7829 034126 022767 000124 143632  5# :    CMP      #124,CPEREG ;IS CPU ERROR REGISTER CORRECT?
7830 034134 001401          BEQ      6#          ;BRANCH IF YES
7831 034136 104001          ERROR    +1          ;CPU ERROR
7832 034140 005067 143622          6# :    CLR      CPEREG     ;CLEAR CPU ERROR REGISTER
7833 034144 012706 001000          MOV      #STBOT,SP    ;RESTORE STACK
7834 034150 016737 146636 000004  MOV      SLOC00,#4     ;RESTORE VECTOR

```

***** DOUBLE OPERAND TESTS *****

```

7835 034156 005037 000000          CLR      0#0          ;RESTORE ADDR 0
7836 034162 005037 000002          CLR      0#2          ;RESTORE ADDR 2
7837
7839
7841
7842 034166          PIRXXX:
7843          ; TEST PIRQ REGISTER RESPONSE
7844 034166 016767 143612 146616      MOV      4,SLOC00      ;SAVE CONTENTS OF VECTORS
7845 034174 012767 034222 143602      MOV      @PIRQNX,4     ;SETUP INTERRUPT VECTOR
7846 034202 005067 143560          CLR      CPREG        ;CLEAR CPU ERROR REGISTER
7847 034206 005737 177772          TST     @PIRQ         ;LOOK FOR REPLY FROM PIRQ
7848 034212 016767 146574 143564      MOV      SLOC00,4     ;RESTORE VECTORS
7849 034220 000401          BR      PIRTEX        ;IF IT RESPONDS, CONTINUE TESTING.
7850          ;ERROR! NO RESPONSE FROM PIRQ REG
7851 034222 104001          PIRQNX: ERROR +1      ;CPU ERROR
7852 034224          PIRTEX:
7853
7854 034224          PIR1:
7855          ; TEST PIRQ REGISTER DATA
7856 034224 013767 000240 146560      MOV      @PIRQVEC,SLOC00 ;SAVE PIRQ VECTORS
7857 034232 013767 000242 146554      MOV      @PIRQVEC+2,SLOC01 ;
7858 034240 012737 034450 000240      MOV      @UNXPIR,@PIRQVEC ;SET UP PIRQ VECTOR FOR UNEXPECTED INTERRUPT
7859 034246 012737 000340 000242      MOV      @PR7,@PIRQVEC+2 ;
7860 034254 012703 001000          MOV      @1000,R3     ;PUT 1000 IN R3: START TESTING
7861          ;BITS IN PIRQ REG BY FLOATING
7862          ;A BIT THROUGH BITS 9-15.
7863 034260 012704 034324          MOV      @PIRTBL,R4   ;SET UP R4 AS A POINTER TO EXPECTED
7864          ;ENCODED PRIORITY LEVELS IN PIRTL
7865 034264 000237          1#: SPL      7        ;DON'T ALLOW INTERRUPTS.
7866 034266 005037 177772          CLR      @PIRQ        ;CLEAR OUT THE PIRQ
7867 034272 023724 177772          CMP      @PIRQ,(R4)+  ;IS PIRQ OK??
7868 034276 001401          BEQ     2#           ;BRANCH IF OK
7869          ;ERROR; PIRQ REG WAS NOT CLEAR
7870 034300 104001          ERROR +1           ;CPU ERROR
7871 034302 010337 177772          2#: MOV      R3,@PIRQ   ;SET A BIT IN PIRQ REGISTER
7872 034306 023724 177772          CMP      @PIRQ,(R4)+ ;COMPARE THE ENCODED PRIORITY BITS
7873          ;WITH DATA IN THE TABLE (PIRTBL)
7874 034312 001401          BEQ     3#           ;BRANCH IF ITS OK
7875          ;ERROR; ENCODED PRIORITY LEVELS ARE
7876          ;NOT CORRECT
7877 034314 104001          ERROR +1           ;CPU ERROR
7878 034316 006303          3#: ASL      R3        ;FLOAT A "1" THRU BITS 9-15
7879 034320 103370          BCC     2#           ;IF CARRY BIT ISN'T SET, DO AGAIN.
7880 034322 000410          BR      EXPIR1       ;GO TO EXIT TEST.
7881
7882 034324 000000          PIRTLBL: .WORD 0
7883 034326 001042          .WORD 1042
7884 034330 002104          .WORD 2104
7885 034332 004146          .WORD 4146
7886 034334 010210          .WORD 10210
7887 034336 020252          .WORD 20252
7888 034340 040314          .WORD 40314
7889 034342 100356          .WORD 100356
7890
7891 034344          EXPIR1:
7892
7902

```

***** DOUBLE OPERAND TESTS *****

```

7903 034344      PIR2:
7904             ; TEST PIRQ REGISTER LEVEL ENCODING
7905             ;*****
;THIS TEST IS TO CHECK THAT THE HIGHEST PRIORITY LEVEL SET IN THE
;PROGRAM INTERRUPT REQUEST BITS IS REFLECTED IN THE ENCODED PROGRAM
;INTERRUPT ACTIVE BITS.

7906 034344 000237      SPL      7      ;SHUT OFF INTERRUPTS
7907 034346 005037 177772      CLR      @PIRQ      ;CLEAR PIRQ REGISTER
7908 034352 012767 000001 146450      MOV      @1,FLAG      ;SETUP END OF LOOP SIGNAL
7909 034360 012703 177000      MOV      @177000,R3      ;SET UP DESIRED PATTERN IN R3
7910 034364 012704 034430      MOV      @PITBL1,R4      ;SETUP R4 AS A TABLE POINTER
7911 034370 010337 177772      1$: MOV      R3,@PIRQ      ;SET PATERN IN PIRQ REGISTER
7912
7913 034374 023724 177772      CMP      @@PIRQ,(R4)+      ;COMPARE PATTERN IN PIRQ WITH PATTERN IN
7914                                     ;EXPECTED PATTERN TABLE.
7915 034400 001012      BNE      2$      ;GO TO ERROR IF NOT THE SAME
7916 034402 005737 003030      TST      @@FLAG      ;IS THIS THE LAST TIME THROUGH??
7917 034406 001421      BEQ      PIR2EX      ;YES, IF FLAG IS ZERO
7918 034410 006003      ROR      R3      ;SHIFT PATTERN TO RIGHT FOR NEXT TEST
7919 034412 042703 000777      BIC      @777,R3      ;STRIP OFF BITS 8-0
7920 034416 001364      BNE      1$      ;IF R3 IS NOT = 0 GO DO THE NEXT PATTERN
7921 034420 005037 003030      CLR      @@FLAG      ;CLR THE FLAG TO INDICATE LAST
7922                                     ;TIME THROUGH THE LOOP
7923 034424 000761      BR       1$      ;GO THROUGH LOOP ONCE MORE
7924                                     ;ERROR; PATTERNS WERE NOT THE SAME
7925 034426 104001      2$: ERROR  +1      ;CPU ERROR
7926
7927 034430 177356 077314 037252 PITBL1: .WORD 177356,77314,37252,17210,7146,3104,1042,0
      034436 017210 007146 003104
      034444 001042 000000

7928
7929 034450      UNXPIR:
7930 034450 104001      ERROR  +1      ;UNEXPECTED PIRQ INTERRUPT
7931                                     ;CPU ERROR
7932 034452      PIR2EX:
7933
7934 034452      PIR3:
7935 034452             ; TEST PIRQ INTERRUPTS
7936             ;*****
;THIS TEST CHECKS THAT EACH PROGRAM INTERRUPT OCCURS PROPERLY

7944 034452 012703 001000      MOV      @1000,R3      ;SETUP R3 AS A WORKING REGISTER
7945 034456 012737 034522 000240      MOV      @PIRRTN,@PIRQVEC      ;SET UP INTERRUPT ROUTINE AT PIR VECTOR
7946 034464 012737 000340 000242      MOV      @340,@PIRQVEC+2      ;
7947 034472 005004      CLR      R4      ;INITIALIZE R4 AS EXPECTED DATA HOLDER.
7948 034474 005724      1$: TST      (R4)+      ;INCREMENT R4 BY 2
7949 034476 050337 177772      BIS      R3,@PIRQ      ;SET PIRQ TO INTERRUPT
7950 034502 000230      SPL      0      ;ENABLE INTERRUPT TO OCCUR
7951                                     ;ERROR! INTERRUPT DID NOT OCCUR
7952 034504 104001      ERROR  +1      ;CPU ERROR
7953                                     ;ERROR! INTERRUPT OCCURRED IN WRONG SEQUENCE
7954 034506 104001      2$: ERROR  +1      ;CPU ERROR
7955 034510 062706 000004      3$: ADD      @4,SP      ;CLEAN UP THE STACK
7956 034514 006303      ASL      R3      ;SHIFT R3 TO LEFT FOR NEXT INTERRUPT
7957 034516 103366      BCC      PI1      ;END IF CARRY BIT IS SET
7958 034520 000412      BR       PIR3EX      ;ON TO THE NEXT TEST
7959
7960 034522 013705 177772      PIRRTN: MOV      @PIRQ,R5      ;MOVE THE CONTENTS OF PIRQ REG TO R5

```

***** DOUBLE OPERAND TESTS *****

```

7961 034526 005037 177772      CLR      @PIRQ      ;KILL PIRQ INTERRUPT.
7962 034532 042705 177761      BIC      @177761,P5 ;MASK OFF ALL BITS EXCEPT 1 4.
7963 034536 020405                CMP      R4,R5      ;DID THE CORRECT LEVEL INTERRUPT OCCUR?
7964 034540 001362                BNE      PI2        ;IF NOT, GO TO ERROR.
7965 034542 000167 177742      JMP      PI3        ;RETURN FROM SUCCESSFUL INTERRUPT.GET
7966                                ;READY FOR THE NEXT ONE.
7967
7968 034546      PIR3EX:
7969
7978
7979 034546      PIR4:
7980      ;      TEST PIRQ VS PSW INTERRUPT LEVEL
7981      ;;*****
;THIS TEST IS TO ENSURE THAT PIRQ CANNOT INTERRUPT WHEN PSW INTERRUPT
;LEVEL IS SET TO BLOCK THEM OUT.
7982 034546 005037 177772      CLR      @PIRQ      ;CLEAR THE PIRQ
7983 034552 005037 177776      CLR      @PSW      ;CLEAR THE PSW.
7984 034556 000237                SPL      7          ;BLOCK INTERRUPTS FROM BEING SERVICED.
7985 034560 012703 001000      MOV      @1000,R3   ;USE R3 AS A WORKING REGISTER.
7986 034564 012737 034616 000240      MOV      @2,@PIRQVEC ;SETUP INTERRUPT VECTORS
7987 034572 012737 000340 000242      MOV      @340,@PIRQVEC+2
7988 034600 010337 177772      1$:      MOV      R3,@PIRQ ;SET INTERRUPT LEVEL IN PIRQ.
7989 034604 006303                ASL      R3          ;SHIFT A "1" LEFT TO INCREASE INTERRUPT
7990                                ;PRIORITY LEVEL.
7991 034606 103374                BCC      1$         ;IF C BIT NOT SET THEN DO IT AGAIN.
7992 034610 005037 177772      CLR      @PIRQ      ;ELSE CLEAR THE PIRQ
7993 034614 000403                BR       3$         ;EXIT TEST.
7994
7995 034616 005037 177772      2$:      CLR      @PIRQ      ;STOP PIRQ FROM INTERRUPTING.
7996                                ;ERROR! NO INTERRUPTS SHOULD OCCUR.
7997 034622 104001                ERROR   +1          ;CPU ERROR
7998 034624      3$:
7999
8013
8014 034624      PIR5:
8015      ;      TEST PIRQ INTERRUPTS OCCUR AT PROPER LEVEL
8016      ;;*****
;THIS TEST ENSURES THAT INTERRUPTS OCCUR AT THE PROPER LEVEL
;THE PRIORITY LEVEL IS INITIALLY SET TO PRI6. THE PIRQ THEN STARTS INTERRUPTING
;AT PRI1. NO INTERRUPTS SHOULD OCCUR UNTIL THE INTERRUPTING LEVEL EXCEEDS THE
;PRIORITY LEVEL IN THE PSW. AFTER EACH LEVEL OF INTERRUPT HAS BEEN TRIED; THE
;PSW PRIORITY LEVEL IS LOWERED ONE NOTCH, AND THE CYCLE REPEATS UNTIL PSW
;PRIORITY LEVEL 0 IS REACHED. INTERRUPTS AT PSW PRIORITY LEVEL 0 HAVE BEEN DONE
;PREVIOUSLY.
8017 034624 005037 177772      CLR      @PIRQ      ;CLR THE PIRQ
8018 034630 012737 034742 000240      MOV      @10,@PIRQVEC ;SET UP INTERRUPT VECTORS
8019 034636 012737 000340 000242      MOV      @PR7,@PIRQVEC+2
8020 034644 012705 035020      MOV      @PITBL2,R5 ;SETUP R5 AS A TABLE POINTER.
8021 034650 012737 000006 114144      MOV      @6,@DCOUNT ;INIT LOCATION DCOUNT:
8022 034656 005004      1$:      CLR      R4          ;INITIALIZE R4 AS A COUNTER.
8023 034660 012703 001000      MOV      @1000,R3   ;USE R3 AS A WORKING REGISTER.
8024 034664 012567 143106      MOV      (R5)+,PS   ;MOVE PRIORITY LEVEL TO PSW
8025 034670 005724      3$:      TST      (R4)+      ;INCREMENT R4 BY 2
8026 034672 010337 177772      MOV      R3,@PIRQ  ;SET INTERRUPT LEVEL IN PIRQ.
8027      ;*****INTERRUPTS WILL HAPPEN HERE*****
8028 034676 013701 177776      MOV      @PS,R1     ;SAVE PS IN R1          !ONLY EXECUTED
8029 034702 013702 177772      MOV      @PIRQ,R2   ;SAVE PIRQ IN R2      ! IF NO

```

***** DOUBLE OPERAND TESTS *****

```

8030 034706 042701 177437      BIC      @177437,R1      ;CLEAR EXTRANEIOUS BITS : INTERRUPT
8031 034712 042702 177437      BIC      @177437,R2      ;                               : HAS
8032 034716 020102              CMP      R1,R2          ;R1 SHOULD BE >= R2      : OCCURRED
8033 034720 002001              BGE      4$             ;IF IT IS GO ON         : --
8034                               ;ERROR! SHOULD HAVE INTEPRUPTED.
8035 034722 104001              ERROR    +1             ;CPU ERROR
8036 034724 006303      4$:    ASL      R3             ;SHJFT A '1" LEFT UNTIL INTERRUPT LEVEL
8037                               ;IS REACHED.
8038 034726 103360              BCC      3$             ;BRANCH BACK IF CARRY IS NOT SET.
8039 034730 005337 114144      DEC      @@DCOUNT      ;LOWER INTERRUPT PRIORITY LEVEL
8040 034734 001350              BNE      1$             ;IF DCOUNT= 0 , WE'RE DONE.
8041 034736 000167 000072      JMP      PIR5EX        ;JUMP TO END OF THIS TEST.
8042
8043      ;PIRQ INTERRUPT SERVICE ROUTINE
8044      ;
8045
8046 034742 013746 177772      10$:    MOV      @@PIRQ,-(SP)      ;SAVE PIRQ DATA ON STACK
8047 034746 005037 177772      CLR      @@PIRQ        ;SHUT OFF PIRQ INTERRUPTS
8048 034752 016601 000004      MOV      4(SP),R1      ;GET OLD PSW. SAVE IN R1.
8049 034756 011602              MOV      (SP),R2       ;GET PIRQ FROM STACK.
8050 034760 042702 177437      BIC      @177437,R2    ;CLEAR UNWANTED BITS FROM R2
8051 034764 042701 177437      BIC      @177437,R1    ;CLEAR UNWANTED BITS FROM R1
8052 034770 020102              CMP      R1,R2         ;R2 SHOULD BE > R1.
8053 034772 100401              BMI      20$          ;GO CHECK SEQUENCE OF INTERRUPT.
8054
8055                               ;ERROR! PRIORITY OF INTERRUPT WAS NOT
8056 034774 104001              ERROR    +1             ;HIGH ENOUGH. SHOULDN'T HAVE OCCURRED.
8057 034776 012602      20$:    MOV      (SP)+,R2      ;CPU ERROR
8058 035000 042702 177761      BIC      @177761,R2    ;POP OLD PIRQ OFF THE STACK.
8059 035004 020402              CMP      R4,R2         ;CLEAR OFF EXTRANEIOUS BITS.
8060 035006 001401              BEQ      21$          ;SHOULD BE EQUAL.
8061                               ;IF THEY ARE EQUAL, CLEAN UP STACK AND
8062                               ;GET READY FOR THE NEXT INTERRUPT
8063                               ;ELSE
8064 035010 104001              ERROR    +1             ;ERROR! INTERRUPT OCCURRED OUT OF SEQUENCE.
8065 035012 012716 034724      21$:    MOV      @4$, (SP)    ;CPU ERROR
8066 035016 000002              RTI                    ;PUT RETURN ADDRESS ON THE STACK.
8067                               ;RETURN FROM INTERRUPT. RESTORE PSW.
8068 035020 000300      PITBL2: .WORD    300      ;PRIORITY LFVEL 6
8069 035022 000240              .WORD    240          ;PRIORITY LEVEL 5
8070 035024 000200              .WORD    200          ;PRIORITY LEVEL 4
8071 035026 000140              .WORD    140          ;PRIORITY LEVEL 3
8072 035030 000100              .WORD    100          ;PRIORITY LEVEL 2
8073 035032 000040              .WORD    40           ;PRIORITY LEVEL 1
8074
8075 035034      PIR5EX:
8076
8091
8092 035034      PIR6:
8093      ; TEST THAT PIRQS ARE SERVICED IN CORRECT ORDER
8094      ; *****
      ;THIS TEST CHECKS THAT ALL PIRQ INTERRUPTS ARE SERVICED
      ;IN THE CORRECT DECENDING ORDER. I.E. IRQ6 IS NOT SERVICED
      ;BEFORE IRQ7 ETC. THE PIRQ IS LOADED WITH ALL INTERRUPTS SIMULTANEOUSLY
      ;TURNED ON. THE PSW PRIORITY LEVEL IS THEN LOWERED TO 0. EACH INTERRUPT
      ;SHOULD BE SERVICED IN DECENDING ORDER. EACH TIME AN INTERRUPT OCCURRS, THE PSW
      ;IS LOADED WITH PRIORITY LEVEL 7 WHICH STOPS THE PIRQ FROM FURTHER INTERRUPTS.

```


MEMORY MANAGEMENT TESTS

8156	035236					
8157						
8158	035236	005067	142524			
8159	035242	005037	177572			
8160	035246	005037	003030			
8161	035252	013746	000244			
8162	035256	013746	000246			
8163	035262	013746	000004			
8164	035266	012737	000246	000244		
8165	035274	012737	000002	000246		
8166	035302	012737	137720	000004		
8167	035310	005005				
8168	035312	012700	172200			
8169	035316	005020				
8170	035320	020027	172400			
8171	035324	001374				
8172	035326	012700	177600			
8173	035332	005020				
8174	035334	020027	177700			
8175	035340	001374				
8176	035342	170127	000200			
8177	035346	012700	003052			
8178	035352	005020				
8179	035354	005020				
8180	035356	005020				
8181	035360	005020				
8182	035362	012700	003052			
8183	035366	172410				
8184	035370	172510				
8185	035372	172610				
8186	035374	172710				
8187	035376	174004				
8188	035400	174005				
8189	035402	174500				
8190	035404	170337	003062			
8191	035410	012704	003062			
8192	035414	022427	000004			
8193	035420	001401				
8194	035422	104002				
8195						
8196	035424	021427	035402			
8197	035430	001401				
8198	035432	104002				
8199						
8200	035434	012704	172200			
8201	035440	012701	000001			
8202	035444	010102				
8203	035446	072227	000010			
8204	035452	021427	000000			
8205	035456	001401				
8206	035460	104002				
8207						
8208	035462	010224				
8209	035464	005201				
8210	035466	020427	172400			
8211	035472	001364				
8212	035474	012704	177600			

```

TSMU2:
; ADDRESS TEST OF PARS,PDRS, AND FP REGS
CLR CPEREG ;CLEAR CPU ERROR REGISTER
CLR @177572 ;MMU OFF
CLR @FLAG ;CLEAR MMU TRAP FLAG
MOV @244,-(SP) ;SAVE FP VECTOR
MOV @246,(SP)
MOV @4,-(SP) ;SAVE TIME OUT VECTOR
MOV @246,@244 ;SETUP NEW FP VECTOR
MOV @ADDRP,@4
CLR R5 ;SETUP NEW TIME OUT VECTOR
MOV @172200,R0 ;CLEAR TIMEOUT FLAG
1$: CLR (R0)+ ;LOAD ALL PARS AND PDRS WITH ZERO
CMP R0,@172400
BNE 1$
MOV @177600,R0
2$: CLR (R0)+
CMP R0,@177700
BNE 2$
LDFPS @200
MOV @FLOAT,R0 ;LOAD ACO-AC5 WITH 0
CLR (R0)+
CLR (R0)+
CLR (R0)+
CLR (R0)+
MOV @FLOAT,R0
LDD (R0),ACO
LDD (R0),AC1
LDD (R0),AC2
LDD (R0),AC3
STD ACO,AC4
STD ACO,AC5
3$: DIVD ACO,AC1 ;LOAD FEC WITH 4 AND FEA WITH #3$
STST @FLO ;CHECK FEC FOR 4 AND FEA FOR #3$
MOV @FLO,R4
CMP (R4)+,@4
BEQ 21$
ERROR +2 ;MMU ERROR
21$: CMP (R4),@3$
BEQ 22$
ERROR +2 ;MMU ERROR
22$: MOV @172200,R4 ;CHECK EACH PAR, PDR FOR 0 THEN
MOV @1,R1 ;WRITE A UNIQUE NUMBER TO IT
4$: MOV R1,R2
ASH #10,R2
CMP (R4),#0
BEQ 5$
ERROR +2 ;MMU ERROR
5$: MOV R2,(R4)+
INC R1
CMP R4,@172400
BNE 4$
MOV @177600,R4
    
```


MEMORY MANAGEMENT TESTS

```

8213 035500 010102      6$:  MOV    R1,R2      ;
8214 035502 072227 000010  ASH    #10,R2      ;
8215 035506 021427 000000  CMP    (R4),#0     ;
8216 035512 001401      BEQ    7$          ;
8217 035514 104002      ERROR  +2          ;MMU ERROR
8218
8219 035516 010224      7$:  MOV    R2,(R4)+   ;
8220 035520 005201      INC    R1          ;
8221 035522 020427 177700  CMP    R4,#177700 ;
8222 035526 001364      BNE    6$          ;
8223 035530 012704 003062  MOV    #FLO,R4     ;CHECK AC5 FOR ALL ZEROES THEN LOAD A 6
8224 035534 012703 003052  MOV    #FLOAT,R3  ;
8225 035540 174014      STD    AC0,(R4)   ;
8226 035542 172405      LDD    AC5,AC0    ;
8227 035544 174013      STD    AC0,(R3)   ;
8228 035546 012702 000004  MOV    #4,R2      ;
8229 035552 022327 000000  8$:  CMP    (R3)+,#0   ;
8230 035556 001401      BEQ    9$          ;
8231 035560 104002      ERROR  +2          ;MMU ERROR
8232
8233 035562 005302      9$:  DEC    R2          ;
8234 035564 001372      BNE    8$          ;
8235 035566 012703 003052  MOV    #FLOAT,R3  ;
8236 035572 012713 000006  MOV    #6,(R3)    ;
8237 035576 172413      LDD    (R3),AC0   ;
8238 035600 174005      STD    AC0,AC5    ;
8239 035602 172404      LDD    AC4,AC0    ;CHECK AC4 FOR ALL ZEROES THEN LOAD A 5
8240 035604 174013      STD    AC0,(R3)   ;
8241 035606 012702 000004  MOV    #4,R2      ;
8242 035612 022327 000000  10$: CMP    (R3)+,#0   ;
8243 035616 001401      BEQ    11$         ;
8244 035620 104002      ERROR  +2          ;MMU ERROR
8245
8246 035622 005302      11$: DEC    R2          ;
8247 035624 001372      BNE    10$        ;
8248 035626 012703 003052  MOV    #FLOAT,R3  ;
8249 035632 012713 000005  MOV    #5,(R3)    ;
8250 035636 172413      LDD    (R3),AC0   ;
8251 035640 174004      STD    AC0,AC4    ;
8252 035642 012702 000004  MOV    #4,R2      ;CHECK AC0 FOR ALL ZEROES THEN LOAD A 1
8253 035646 022427 000000  12$: CMP    (R4)+,#0   ;
8254 035652 001401      BEQ    13$         ;
8255 035654 104002      ERROR  +2          ;MMU ERROR
8256
8257 035656 005302      13$: DEC    R2          ;
8258 035660 001372      BNE    12$        ;
8259 035662 012713 000001  MOV    #1,(R3)    ;
8260 035666 172413      LDD    (R3),AC0   ;
8261 035670 012704 003062  MOV    #FLO,R4     ;CHECK AC1 FOR ALL ZEROES THEN LOAD A 2
8262 035674 012702 000004  MOV    #4,R2      ;
8263 035700 174114      STD    AC1,(R4)   ;
8264 035702 022427 000000  14$: CMP    (R4)+,#0   ;
8265 035706 001401      BEQ    15$         ;
8266 035710 104002      ERROR  +2          ;MMU ERROR
8267
8268 035712 005302      15$: DEC    R2          ;
8269 035714 001372      BNE    14$        ;

```

MEMORY MANAGEMENT TESTS

8270	035716	012713	000002	MOV	#2,(R3)	:
8271	035722	172513		LDD	(R3),AC1	:
8272	035724	012704	003062	MOV	#FLO,R4	;CHECK AC2 FOR ALL ZEROES THEN LOAD A 3
8273	035730	012702	000004	MOV	#4,R2	:
8274	035734	174214		STD	AC2,(R4)	:
8275	035736	022427	000000	16\$: CMP	(R4)+,#0	:
8276	035742	001401		BEQ	17\$:
8277	035744	104002		ERROR	+2	;MMU ERROR
8278						:
8279	035746	005302		17\$: DEC	R2	:
8280	035750	001372		BNE	16\$:
8281	035752	012713	000003	MOV	#3,(R3)	:
8282	035756	172613		LDD	(R3),AC2	:
8283	035760	012704	003062	MOV	#FLO,R4	;CHECK AC3 FOR ALL ZEROES THEN LOAD A 4
8284	035764	012702	000004	MOV	#4,R2	:
8285	035770	174314		STD	AC3,(R4)	:
8286	035772	022427	000000	18\$: CMP	(R4)+,#0	:
8287	035776	001401		BEQ	19\$:
8288	036000	104002		ERROR	+2	;MMU ERROR
8289						:
8290	036002	005302		19\$: DEC	R2	:
8291	036004	001372		BNE	18\$:
8292	036006	012713	000004	MOV	#4,(R3)	:
8293	036012	172713		LDD	(R3),AC3	:
8294	036014	012704	003062	MOV	#FLO,R4	;CHECK FPS FOR 100204 THEN LOAD IT WITH 200
8295	036020	170214		STFPS	(R4)	:
8296	036022	022714	100204	CMP	#100204,(R4)	:
8297	036026	001401		BEQ	20\$:
8298	036030	104002		ERROR	+2	;MMU ERROR
8299						:
8300	036032	170127	000200	20\$: LDFPS	#200	:
8301	036036	012704	172200	MOV	#172200,R4	;CHECK PDR, PAR FOR UNIQUE NUMBERS
8302	036042	012701	000001	MOV	#1,R1	:
8303	036046	010102		23\$: MOV	R1,R2	:
8304	036050	072227	000010	ASH	#10,R2	:
8305	036054	022402		CMP	(R4)+,R2	:
8306	036056	001401		BEQ	24\$:
8307	036060	104002		ERROR	+2	;MMU ERROR
8308						:
8309	036062	005201		24\$: INC	R1	:
8310	036064	020427	172400	CMP	R4,#172400	:
8311	036070	001366		BNE	23\$:
8312	036072	012704	177600	MOV	#177600,R4	:
8313	036076	010102		25\$: MOV	R1,R2	:
8314	036100	072227	000010	ASH	#10,R2	:
8315	036104	022402		CMP	(R4)+,R2	:
8316	036106	001401		BEQ	26\$:
8317	036110	104002		ERROR	+2	;MMU ERROR
8318						:
8319	036112	005201		26\$: INC	R1	:
8320	036114	020427	177700	CMP	R4,#177700	:
8321	036120	001366		BNE	25\$:
8322	036122	012701	003052	MOV	#FLOAT,R1	;CHECK ACS FOR #6
8323	036126	012704	003062	MOV	#FLO,R4	:
8324	036132	174014		STD	AC0,(R4)	:
8325	036134	172405		LDD	AC5,AC0	:
8326	036136	174011		STD	AC0,(R1)	:

MEMORY MANAGEMENT TESTS

8327	036140	022127	000006		CMP	(R1)+, #6		:
8328	036144	001401			BEQ	27\$:
8329	036146	104002			ERROR	+2		;MMU ERROR
8330								:
8331	036150	012703	000003	27\$:	MOV	#3,R3		:
8332	036154	022127	000000	28\$:	CMP	(R1)+, #0		:
8333	036160	001401			BEQ	29\$:
8334	036162	104002			ERROR	+2		;MMU ERROR
8335								:
8336	036164	005303		29\$:	DEC	R3		:
8337	036166	001372			BNE	28\$:
8338	036170	012701	003052		MOV	#FLOAT,R1		;CHECK AC4 FOR #5
8339	036174	172404			LDD	AC4,AC0		:
8340	036176	174011			STD	AC0,(R1)		:
8341	036200	022127	000005		CMP	(R1)+, #5		:
8342	036204	001401			BEQ	30\$:
8343	036206	104002			ERROR	+2		;MMU ERROR
8344								:
8345	036210	012703	000003	30\$:	MOV	#3,R3		:
8346	036214	022127	000000	31\$:	CMP	(R1)+, #0		:
8347	036220	001401			BEQ	32\$:
8348	036222	104002			ERROR	+2		;MMU ERROR
8349								:
8350	036224	005303		32\$:	DEC	R3		:
8351	036226	001372			BNE	31\$:
8352	036230	022427	000001		CMP	(R4)+, #1		;CHECK AC0 FOR #1
8353	036234	001401			BEQ	33\$:
8354	036236	104002			ERROR	+2		;MMU ERROR
8355								:
8356	036240	012703	000003	33\$:	MOV	#3,R3		:
8357	036244	022427	000000	34\$:	CMP	(R4)+, #0		:
8358	036250	001401			BEQ	35\$:
8359	036252	104002			ERROR	+2		;MMU ERROR
8360								:
8361	036254	005303		35\$:	DEC	R3		:
8362	036256	001372			BNE	34\$:
8363	036260	012701	003052		MOV	#FLOAT,R1		;CHECK AC1 FOR #2
8364	036264	174111			STD	AC1,(R1)		:
8365	036266	022127	000002		CMP	(R1)+, #2		:
8366	036272	001401			BEQ	36\$:
8367	036274	104002			ERROR	+2		;MMU ERROR
8368								:
8369	036276	012703	000003	36\$:	MOV	#3,R3		:
8370	036302	022127	000000	37\$:	CMP	(R1)+, #0		:
8371	036306	001401			BEQ	38\$:
8372	036310	104002			ERROR	+2		;MMU ERROR
8373								:
8374	036312	005303		38\$:	DEC	R3		:
8375	036314	001372			BNE	37\$:
8376	036316	012701	003052		MOV	#FLOAT,R1		;CHECK AC2 FOR #3
8377	036322	174211			STD	AC2,(R1)		:
8378	036324	022127	000003		CMP	(R1)+, #3		:
8379	036330	001401			BEQ	39\$:
8380	036332	104002			ERROR	+2		;MMU ERROR
8381								:
8382	036334	012703	000003	39\$:	MOV	#3,R3		:
8383	036340	022127	000000	40\$:	CMP	(R1)+, #0		:

MEMORY MANAGEMENT TESTS

```

8384 036344 001401      BEQ      41$
8385 036346 104002      ERROR    +2          ;MMU ERROR
8386
8387 036350 005303      41$:    DEC      R3
8388 036352 001372      BNE     40$
8389 036354 012701 003052    MOV     @FLOAT,R1    ;CHECK AC3 FOR #4
8390 036360 174311      STD     AC3,(R1)
8391 036362 022127 000004    CMP     (R1)+,#4
8392 036366 001401      BEQ     42$
8393 036370 104002      ERROR    +2          ;MMU ERROR
8394
8395 036372 012703 000003    42$:    MOV     #3,R3
8396 036376 022127 000000    43$:    CMP     (R1)+,#0
8397 036402 001401      BEQ     44$
8398 036404 104002      ERROR    +2          ;MMU ERROR
8399
8400 036406 005303      44$:    DEC      R3
8401 036410 001372      BNE     43$
8402 036412 020527 000000    CMP     R5,#0        ;IS TIME OUT FLAG 0
8403 036416 001401      BEQ     45$        ;YES GO ON
8404 036420 104002      ERROR    +2          ;MMU ERROR
8405
8406 036422 012637 000004    45$:    MOV     (SP)+,#04    ;RESTORE TIME OUT VECTOR
8407 036426 012637 000246    MOV     (SP)+,#0246 ;RESTORE FP VECTOR
8408 036432 012637 000244    MOV     (SP)+,#0244
8409
8412 036436      TSMMU3:
8413      ; WRITE ALL PARS/PDRS WITH ONES THEN ZEROS
8414 036436 005037 177572    CLR     @177572      ;MMU OFF
8415 036442 005037 003030    CLR     @FLAG        ;CLEAR MMU ABORT FLAG
8416 036446 012703 172200    MOV     @172200,R3   ;LOAD ALL PARS AND PDRS WITH ONES
8417 036452 012723 177777    1$:    MOV     @177777,(R3)+
8418 036456 020327 172400    CMP     R3,#172400
8419 036462 001373      BNE     1$
8420 036464 012703 177600    MOV     @177600,R3
8421 036470 012723 177777    2$:    MOV     @177777,(R3)+
8422 036474 020327 177700    CMP     R3,#177700
8423 036500 001373      BNE     2$
8424 036502 012703 172200    MOV     @172200,R3   ;CHECK SPDRS FOR ONES
8425 036506 022327 177416    3$:    CMP     (R3)+,#177416
8426 036512 001401      BEQ     4$
8427 036514 104002      ERROR    +2          ;MMU ERROR
8428
8429 036516 020327 172240    4$:    CMP     R3,#172240
8430 036522 001371      BNE     3$
8431 036524 022327 177777    5$:    CMP     (R3)+,#177777 ;CHECK SPARS FOR ONES
8432 036530 001401      BEQ     6$
8433 036532 104002      ERROR    +2          ;MMU ERROR
8434
8435 036534 020327 172300    6$:    CMP     R3,#172300
8436 036540 001371      BNE     5$
8437 036542 022327 177416    7$:    CMP     (R3)+,#177416 ;CHECK KPDRS FOR ONES
8438 036546 001401      BEQ     8$
8439 036550 104002      ERROR    +2          ;MMU ERROR
8440
8441 036552 020327 172340    8$:    CMP     R3,#172340
8442 036556 001371      BNE     7$

```

MEMORY MANAGEMENT TESTS

```

8443 036560 022327 177777      9$:  CMP      (R3)+, #177777      ;CHECK KPARS FOR ONES
8444 036564 001401              BEQ      10$                  ;
8445 036566 104002              ERROR    +2                  ;MMU ERROR
8446                               ;
8447 036570 020327 172400      10$:  CMP      R3, #172400        ;
8448 036574 001371              BNE     9$                   ;
8449 036576 012703 177600      MOV     #177600, R3          ;CHECK UPDRS FOR ONES
8450 036602 022327 177416      11$:  CMP      (R3)+, #177416    ;
8451 036606 001401              BEQ     12$                  ;
8452 036610 104002              ERROR    +2                  ;MMU ERROR
8453                               ;
8454 036612 020327 177640      12$:  CMP      R3, #177640        ;
8455 036616 001371              BNE     11$                 ;
8456 036620 022327 177777      13$:  CMP      (R3)+, #177777    ;CHECK UPARS FOR ONES
8457 036624 001401              BEQ     14$                  ;
8458 036626 104002              ERROR    +2                  ;MMU ERROR
8459                               ;
8460 036630 020327 177700      14$:  CMP      R3, #177700        ;
8461 036634 001371              BNE     13$                 ;
8462 036636 012703 172200      MOV     #172200, R3          ;LOAD ALL PARS AND PDRS WITH ZEROES
8463 036642 012723 000000      15$:  MOV     #0, (R3)+         ;
8464 036646 020327 172400      CMP     R3, #172400         ;
8465 036652 001373              BNE     15$                 ;
8466 036654 012703 177600      MOV     #177600, R3          ;
8467 036660 012723 000000      16$:  MOV     #0, (R3)+         ;
8468 036664 020327 177700      CMP     R3, #177700         ;
8469 036670 001373              BNE     16$                 ;
8470 036672 012703 172200      MOV     #172200, R3          ;CHECK ALL PARS AND PDRS FOR ZERES
8471 036676 022327 000000      17$:  CMP     (R3)+, #0         ;
8472 036702 001401              BEQ     18$                  ;
8473 036704 104002              ERROR    +2                  ;MMU ERROR
8474                               ;
8475 036706 020327 172400      18$:  CMP     R3, #172400        ;
8476 036712 001371              BNE     17$                 ;
8477 036714 012703 177600      MOV     #177600, R3          ;
8478 036720 022327 000000      19$:  CMP     (R3)+, #0         ;
8479 036724 001401              BEQ     20$                  ;
8480 036726 104002              ERROR    +2                  ;MMU ERROR
8481                               ;
8482 036730 020327 177700      20$:  CMP     R3, #177700        ;
8483 036734 001371              BNE     19$                 ;
8484                               ;
8487 036736                      TSMU4:
8488                               ;
8489 036736 005037 177572      ; TEST FOR ADJACENT SHORTS IN PARS/PDRS
8490 036742 005067 144062      CLR     #177572             ;MMU OFF
8491 036746 012700 172200      CLR     FLAG                ;CLEAR MMU ABORT FLAG
8492 036752 012720 052404      1$:  MOV     #172200, R0         ;LOAD SPDRS WITH ALTERNATING PATTERN
8493 036756 012720 125012      MOV     #52404, (R0)+      ;
8494 036762 020027 172240      MOV     #125012, (R0)+    ;
8495 036766 001371              CMP     R0, #172240        ;
8496 036770 012720 125252      2$:  BNE     1$                 ;
8497 036774 012720 052525      MOV     #125252, (R0)+    ;LOAD SPARS WITH ALTERNATING PATTERN
8498 037000 020027 172300      MOV     #52525, (R0)+    ;
8499 037004 001371              CMP     R0, #172300        ;
8500 037006 012720 052404      3$:  BNE     2$                 ;
8501 037012 012720 125012      MOV     #52404, (R0)+    ;LOAD KPDRS WITH ALTERNATING PATTERN
8501 037012 012720 125012      MOV     #125012, (R0)+    ;

```

MEMORY MANAGEMENT TFSTS

8502	037016	020027	172340		CMP	R0,#172340	;
8503	037022	001371			BNE	3#	;
8504	037024	012720	125252	4#:	MOV	#125252,(R0)+	;LOAD KPARS WITH ALTERNATING PATTERN
8505	037030	012720	052525		MOV	#52525,(R0)+	;
8506	037034	020027	172400		CMP	R0,#172400	;
8507	037040	001371			BNE	4#	;
8508	037042	012700	177600		MOV	#177600,R0	;LOAD UPDRS WITH ALTERNATING PATTERN
8509	037046	012720	052404	5#:	MOV	#52404,(R0)+	;
8510	037052	012720	125012		MOV	#125012,(R0)+	;
8511	037056	020027	177640		CMP	R0,#177640	;
8512	037062	001371			BNE	5#	;
8513	037064	012720	125252	6#:	MOV	#125252,(R0)+	;LOAD UPARS WITH ALTERNATING PATTERN
8514	037070	012720	052525		MOV	#52525,(R0)+	;
8515	037074	020027	177700		CMP	R0,#177700	;
8516	037100	001371			BNE	6#	;
8517							;
8518	037102	012703	172200		MOV	#172200,R3	;CHECK SPDRS
8519	037106	022327	052404	7#:	CMP	(R3)+,#52404	;
8520	037112	001401			BEQ	8#	;
8521	037114	104002			ERROR	+2	;MMU ERROR
8522							;
8523	037116	022327	125012	8#:	CMP	(R3)+,#125012	;
8524	037122	001401			BEQ	9#	;
8525	037124	104002			ERROR	+2	;MMU ERROR
8526							;
8527	037126	020327	172240	9#:	CMP	R3,#172240	;
8528	037132	001365			BNE	7#	;
8529	037134	022327	125252	10#:	CMP	(R3)+,#125252	;CHECK SPARS
8530	037140	001401			BEQ	11#	;
8531	037142	104002			ERROR	+2	;MMU ERROR
8532							;
8533	037144	022327	052525	11#:	CMP	(R3)+,#52525	;
8534	037150	001401			BEQ	12#	;
8535	037152	104002			ERROR	+2	;MMU ERROR
8536							;
8537	037154	020327	172300	12#:	CMP	R3,#172300	;
8538	037160	001365			BNE	10#	;
8539	037162	022327	052404	13#:	CMP	(R3)+,#52404	;CHECK KPDRS
8540	037166	001401			BEQ	14#	;
8541	037170	104002			ERROR	+2	;MMU ERROR
8542							;
8543	037172	022327	125012	14#:	CMP	(R3)+,#125012	;
8544	037176	001401			BEQ	15#	;
8545	037200	104002			ERROR	+2	;MMU ERROR
8546							;
8547	037202	020327	172340	15#:	CMP	R3,#172340	;
8548	037206	001365			BNE	13#	;
8549	037210	022327	125252	16#:	CMP	(R3)+,#125252	;CHECK KPARS
8550	037214	001401			BEQ	17#	;
8551	037216	104002			ERROR	+2	;MMU ERROR
8552							;
8553	037220	022327	052525	17#:	CMP	(R3)+,#52525	;
8554	037224	001401			BEQ	18#	;
8555	037226	104002			ERROR	+2	;MMU ERROR
8556							;
8557	037230	020327	172400	18#:	CMP	R3,#172400	;
8558	037234	001365			BNE	16#	;

MEMORY MANAGEMENT TESTS

```

8559 037236 012703 177600      MOV      #177600,R3      ;CHECK UPDRS
8560 037242 022327 052404      19#:    CMP      (R3)+,#52404      ;
8561 037246 001401              BEQ      20#              ;
8562 037250 104002              ERROR    +2              ;MMU ERROR
8563                                ;
8564 037252 022327 125012      20#:    CMP      (R3)+,#125012     ;
8565 037256 001401              BEQ      21#              ;
8566 037260 104002              ERROR    +2              ;MMU ERROR
8567                                ;
8568 037262 020327 177640      21#:    CMP      R3,#177640        ;
8569 037266 001365              BNE     19#              ;
8570 037270 022327 125252      22#:    CMP      (R3)+,#125252     ;CHECK UPARS
8571 037274 001401              BEQ      23#              ;
8572 037276 104002              ERROR    +2              ;MMU ERROR
8573                                ;
8574 037300 022327 052525      23#:    CMP      (R3)+,#52525     ;
8575 037304 001401              BEQ      24#              ;
8576 037306 104002              ERROR    +2              ;MMU ERROR
8577                                ;
8578 037310 020327 177700      24#:    CMP      R3,#177700        ;
8579 037314 001365              BNE     22#              ;
8580                                ;
8581                                ;REVERSE ALTERNATING PATTERN
8582                                ;
8583 037316 012700 172200      MOV      #172200,R0      ;LOAD SPDRS WITH REVERSE PATTERN
8584 037322 012720 125012      25#:    MOV      #125012,(R0)+     ;
8585 037326 012720 052404      MOV      #52404,(R0)+     ;
8586 037332 020027 172240      CMP      R0,#172240      ;
8587 037336 001371              BNE     25#              ;
8588 037340 012720 052525      26#:    MOV      #52525,(R0)+     ;LOAD SPARS WITH REVERSE PATTERN
8589 037344 012720 125252      MOV      #125252,(R0)+     ;
8590 037350 020027 172300      CMP      R0,#172300      ;
8591 037354 001371              BNE     26#              ;
8592 037356 012720 125012      27#:    MOV      #125012,(R0)+     ;LOAD KPDRS WITH REVERSE PATTERN
8593 037362 012720 052404      MOV      #52404,(R0)+     ;
8594 037366 020027 172340      CMP      R0,#172340      ;
8595 037372 001371              BNE     27#              ;
8596 037374 012720 052525      28#:    MOV      #52525,(R0)+     ;LOAD KPARS WITH REVERSE PATTERN
8597 037400 012720 125252      MOV      #125252,(R0)+     ;
8598 037404 020027 172400      CMP      R0,#172400      ;
8599 037410 001371              BNE     28#              ;
8600 037412 012700 177600      MOV      #177600,R0      ;LOAD UPDRS WITH REVERSE PATTERN
8601 037416 012720 125012      29#:    MOV      #125012,(R0)+     ;
8602 037422 012720 052404      MOV      #52404,(R0)+     ;
8603 037426 020027 177640      CMP      R0,#177640      ;
8604 037432 001371              BNE     29#              ;
8605 037434 012720 052525      30#:    MOV      #52525,(R0)+     ;LOAD UPARS WITH REVERSE PATTERN
8606 037440 012720 125252      MOV      #125252,(R0)+     ;
8607 037444 020027 177700      CMP      R0,#177700      ;
8608 037450 001371              BNE     30#              ;
8609                                ;
8610 037452 012703 172200      MOV      #172200,R3      ;CHECK SPDRS
8611 037456 022327 125012      31#:    CMP      (R3)+,#125012     ;
8612 037462 001401              BEQ      32#              ;
8613 037464 104002              ERROR    +2              ;MMU ERROR
8614                                ;
8615 037466 022327 052404      32#:    CMP      (R3)+,#52404     ;

```


MEMORY MANAGEMENT TESTS

```

8675 037666
8676
8677 037666 012737 160000 177572
8678 037674 005067 143130
8679 037700 013700 177572
8680 037704 042700 000176
8681 037710 020027 160000
8682 037714 001401
8683 037716 104002
8684
8685 037720 005037 177572 1$:
8686 037724 013700 177572
8687 037730 042700 000176
8688 037734 020027 000000
8689 037740 001401
8690 037742 104002
8691
8692 037744 012737 120000 177572 2$:
8693 037752 013700 177572
8694 037756 042700 000176
8695 037762 020027 120000
8696 037766 001401
8697 037770 104002
8698
8699 037772 012737 040000 177572 3$:
8700 040000 013700 177572
8701 040004 042700 000176
8702 040010 020027 040000
8703 040014 001401
8704 040016 104002
8705 040020
8708 040020
8709
8710 040020 005037 177572
8711 040024 005067 143000
8712 040030 012737 000077 172516
8713 040036 023727 172516 000077
8714 040044 001401
8715 040046 104002
8716 040050 005037 172516 1$:
8717 040054 023727 172516 000000
8718 040062 001401
8719 040064 104002
8720 040066 012737 000052 172516 2$:
8721 040074 023727 172516 000052
8722 040102 001401
8723 040104 104002
8724 040106 012737 000025 172516 3$:
8725 040114 023727 172516 000025
8726 040122 001401
8727 040124 104002
8728 040126
8731 040126
8732
8733 040126 005037 177572
8734 040132 005037 003030
8735 040136 012737 140000 177776

```

```

TSMU5:
;
TEST MMRO ABORT BITS
MOV #160000,#177572 ;LOAD MMRO<15:13>=111
CLR FLAG ;CLEAR MMU ABORT FLAG
MOV @SRO,R0 ;SAVE SRO IN R0
BIC #176,R0 ;CLEAR UNDEFINED BITS FROM SRO
CMP R0,#160000 ;CHECK MMRO
BEQ 1$
ERROR +2 ;MMU ERROR
;
;
1$: CLR #177572 ;LOAD MMRO=0
MOV @SRO,R0 ;SAVE SRO IN R0
BIC #176,R0 ;CLEAR UNDEFINED BITS FROM SRO
CMP R0,#0 ;CHECK MMRO
BEQ 2$
ERROR +2 ;MMU ERROR
;
;
2$: MOV #120000,#177572 ;LOAD MMRO<15:13>=101
MOV @SRO,R0 ;SAVE SRO IN R0
BIC #176,R0 ;CLEAR UNDEFINED BITS FROM SRO.
CMP R0,#120000 ;CHECK MMRO
BEQ 3$
ERROR +2 ;MMU ERROR
;
;
3$: MOV #40000,#177572 ;LOAD MMRO<15:13>=010
MOV @SRO,R0 ;SAVE SRO IN R0
BIC #176,R0 ;CLEAR UNDEFINED BITS FROM SRO.
CMP R0,#40000 ;CHECK MMRO
BEQ 4$
ERROR +2 ;MMU ERROR
;
;
4$:
TSMU6:
;
TEST MMR3 BITS 5-0
CLR #177572 ;MMU OFF
CLR FLAG ;CLEAR MMU ABORT FLAG
MOV #77,#172516 ;LOAD MMR3<5:0>=77
CMP @172516,#77 ;CHECK MMR3
BEQ 1$
ERROR +2 ;MMU ERROR
;
;
1$: CLR #172516 ;LOAD MMR3<5:0>=0
CMP @172516,#0 ;CHECK MMR3
BEQ 2$
ERROR +2 ;MMU ERROR
;
;
2$: MOV #52,#172516 ;LOAD MMR3<5:0>=52
CMP @172516,#52 ;CHECK MMR3
BEQ 3$
ERROR +2 ;MMU ERROR
;
;
3$: MOV #25,#172516 ;LOAD MMR3<5:0>=25
CMP @172516,#25 ;CHECK MMR3
BEQ 4$
ERROR +2 ;MMU ERROR
;
;
4$:
TSMU6A:
;
TEST MFPI (MOVE FROM PREVIOUS INST SPACE)
CLR #177572 ;MMU OFF
CLR @FLAG ;CLEAR MMU ABORT FLAG
MOV #140000,#177776 ;POINT TO USER SPACE

```


MEMORY MANAGEMENT TESTS

```

8793 040434 023727 000244 173621      CMP      @0244,@173621      ;IS TEST LOCATION CORRECT
8794 040442 001401                      BEQ      301$              ;YES GO ON
8795 040444 104002                      ERROR    +2                ;MMU ERROR
8796                                     ;NO GO TO ERROR
8797 040446 020127 000244      301$:   CMP      R1,@244      ;IS R1 CORRECT
8798 040452 001401                      BEQ      302$              ;YES GO ON
8799 040454 104002                      ERROR    +2                ;MMU ERROR
8800                                     ;NO GO TO ERROR
8801 040456 005037 177776      302$:   CLR      @0177776      ;SET PSW TO KERNEL MODE
8802 040462 022627 173621      CMP      (SP)+,@173621    ;IS STACK CORRECT
8803 040466 001401                      BEQ      303$              ;YES GO ON
8804 040470 104002                      ERROR    +2                ;MMU ERROR
8805                                     ;NO GO TO ERROR
8806 040472 021627 177777      303$:   CMP      (SP),@177777 ;IS STACK CORRECT
8807 040476 001401                      BEQ      304$              ;YES GO ON
8808 040500 104002                      ERROR    +2                ;MMU ERROR
8809                                     ;NO GO TO ERROR
8810 040502 005003                      304$:   CLR      R3          ;SETUP SOURCE FOR NEXT TEST
8811 040504 005237 177572      INC      @0177572        ;TURN MMU ON
8812 040510 006503                      MFPI     R3              ; TEST INSTRUCTION
8813 040512 022737 000004 177776    CMP      @4,@0177776     ;IS PSW CORRECT
8814 040520 001401                      BEQ      6$               ;YES GO ON
8815 040522 104002                      ERROR    +2                ;MMU ERROR
8816                                     ;NO GO TO ERROR
8817 040524 005037 177572      6$:     CLR      @0177572    ;TURN MMU OFF
8818 040530 020327 000000      CMP      R3,@0           ;IS R3 CORRECT
8819 040534 001401                      BEQ      7$               ;YES GO ON
8820 040536 104002                      ERROR    +2                ;MMU ERROR
8821                                     ;NO GO TO ERROR
8822 040540 022627 000000      7$:     CMP      (SP)+,@0      ;IS STACK CORRECT
8823 040544 001401                      BEQ      8$               ;YES GO ON
8824 040546 104002                      ERROR    +2                ;MMU ERROR
8825                                     ;NO GO TO ERROR
8826 040550 022627 177777      8$:     CMP      (SP)+,@177777 ;IS STACK CORRECT
8827 040554 001401                      BEQ      9$               ;YES GO ON
8828 040556 104002                      ERROR    +2                ;MMU ERROR
8829                                     ;NO GO TO ERROR
8830 040560 012637 000244      9$:     MOV      (SP)+,@0244    ;RESTORE TEST LOCATION
8831
8832                                     ;
8835 040564                                     ;TSMM6B:
8836                                     ;
8837 040564 005037 177572      ; TEST MFPD (MOVE FROM PREVIOUS DATA SPACE)
8838 040570 005037 003030      CLR      @0177572        ;MMU OFF
8839 040574 012737 140000 177776    CLR      @0FLAG          ;CLEAR MMU ABORT FLAG
8840 040602 010637 003040      MOV      @140000,@0177776 ;POINT TO USER SPACE
8841 040606 012737 040000 177776    MOV      R6,@0SAVUSE     ;SAVE USER SP
8842 040614 010637 003036      MOV      @40000,@0177776 ;POINT TO SUPERVISOR SPACE
8843 040620 012737 030000 177776    MOV      R6,@0SAVSUP     ;SAVE SUPERVISOR SP
8844 040626 004767 076720      MOV      @30000,@0177776 ;SETUP PSW
8845 040632 012737 000027 172516    JSR      PC,MMU          ;INIT MMU
8846 040640 013746 000244      MOV      @27,@0172516    ;SETUP MMR3
8847 040644 012746 177777      MOV      @0244,-(SP)     ;SAVE DATA AT TEST LOCATION
8848 040650 012737 157002 000244    MOV      @177777,-(SP)   ;PUT KNOWN DATA ON TOP OF STACK
8849 040656 012767 077400 136714    MOV      @157002,@0244   ;SETUP DATA AT TEST LOCATION
8850 040664 012703 000244      MOV      @77400,UIPDR0   ;SETUP UIPDR0 TO ABORT
8851 040670 005237 177572      MOV      @244,R3         ;SETUP POINTER TO TEST LOCATION
                                     INC      @0177572        ;TURN MMU ON

```

MEMORY MANAGEMENT TESTS

8852	040674	106523				MFPD	(R3),		; TEST INSTRUCTION
8853	040676	022737	030010	177776		CMP	#30010,#177776		;IS PSW CORRECT
8854	040704	001401				BEQ	1#		;YES GO ON
8855	040706	104002				ERROR	+2		;MMU ERROR
8856									;NO GO TO ERROR
8857	040710	005037	177572		1#:	CLR	#177572		;TURN MMU OFF
8858	040714	012737	140000	177776		MOV	#140000,#177776		;POINT TO USER SPACE
8859	040722	020637	003040			CMP	R6,#SAVUSE		;IS USER SP CORRECT
8860	040726	001401				BEQ	100#		;YES GO ON
8861	040730	104002				ERROR	+2		;MMU ERROR
8862									;NO GO TO ERROR
8863	040732	012737	040000	177776	100#:	MOV	#40000,#177776		;POINT TO SUPERVISOR SPACE
8864	040740	020637	003036			CMP	R6,#SAVSUP		;IS SUPERVISOR SP CORRECT
8865	040744	001401				BEQ	200#		;YES GO ON
8866	040746	104002				ERROR	+2		;MMU ERROR
8867									;NO GO TO ERROR
8868	040750	023727	000244	157002	200#:	CMP	#244,#157002		;IS TEST DATA OK
8869	040756	001401				BEQ	2#		;YES GO ON
8870	040760	104002				ERROR	+2		;MMU ERROR
8871									;NO GO TO ERROR
8872	040762	020327	000246		2#:	CMP	R3,#246		;IS R3 CORRECT
8873	040766	001401				BEQ	3#		;YES GO ON
8874	040770	104002				ERROR	+2		;MMU ERROR
8875									;NO GO TO ERROR
8876	040772	005037	177776		3#:	CLR	#177776		;SET PSW TO KERNEL MODE
8877	040776	022627	157002			CMP	(SP),#157002		;IS KERNEL STACK CORRECT
8878	041002	001401				BEQ	4#		;YES GO ON
8879	041004	104002				ERROR	+2		;MMU ERROR
8880									;NO GO TO ERROR
8881	041006	021627	177777		4#:	CMP	(SP),#177777		;IS STACK CORRECT
8882	041012	001401				BEQ	5#		;YES GO ON
8883	041014	104002				ERROR	+2		;MMU ERROR
8884									;NO GO TO ERROR
8885	041016	012737	030017	177776	5#:	MOV	#30017,#177776		;SETUP PSW
8886	041024	012737	103456	000244		MOV	#103456,#244		;SETUP TEST LOCATION
8887	041032	012701	000244			MOV	#244,R1		;SETUP R1
8888	041036	005237	177572			INC	#177572		;TURN MMU ON
8889	041042	106511				MFPD	(R1)		;TEST INSTRUCTION
8890	041044	022737	030011	177776		CMP	#30011,#177776		;IS PSW CORRECT
8891	041052	001401				BEQ	300#		;YES GO ON
8892	041054	104002				ERROR	+2		;MMU ERROR
8893									;NO GO TO ERROR
8894	041056	005037	177572		300#:	CLR	#177572		;TURN MMU OFF
8895	041062	023727	000244	103456		CMP	#244,#103456		;IS TEST LOCATION CORRECT
8896	041070	001401				BEQ	301#		;YES GO ON
8897	041072	104002				ERROR	+2		;MMU ERROR
8898									;NO GO TO ERROR
8899	041074	020127	000244		301#:	CMP	R1,#244		;IS R1 CORRECT
8900	041100	001401				BEQ	302#		;YES GO ON
8901	041102	104002				ERROR	+2		;MMU ERROR
8902									;NO GO TO ERROR
8903	041104	005037	177776		302#:	CLR	#177776		;SET PSW TO KERNEL MODE
8904	041110	022627	103456			CMP	(SP),#103456		;IS STACK CORRECT
8905	041114	001401				BEQ	303#		;YES GO ON
8906	041116	104002				ERROR	+2		;MMU ERROR
8907									;NO GO TO ERROR
8908	041120	021627	177777		303#:	CMP	(SP),#177777		;IS STACK CORRECT

MEMORY MANAGEMENT TESTS

```

8909 041124 001401          BEQ      304$          ;YES GO ON
8910 041126 104002          ERROR    +2           ;MMU ERROR
8911                               ;NO GO TO ERROR
8912 041130 012737 030017 177776 304$:  MOV     @30017,@177776 ;SETUP PSW
8913 041136 012737 113672 000244      MOV     @113672,@244   ;SETUP TEST LOCATION
8914 041144 012701 000246      MOV     @246,R1       ;SETUP R1
8915 041150 005237 177572      INC     @177572       ;TURN MMU ON
8916 041154 106541          MFPD   -(R1)         ;TEST INSTRUCTION
8917 041156 022737 030011 177776      CMP     @30011,@177776 ;IS PSW CORRECT
8918 041164 001401          BEQ     400$          ;YES GO ON
8919 041166 104002          ERROR    +2           ;MMU ERROR
8920                               ;NO GO TO ERROR
8921 041170 005037 177572          CLR     @177572       ;TURN MMU OFF
8922 041174 023727 000244 113672      CMP     @244,@113672  ;IS TEST LOCATION CORRECT
8923 041202 001401          BEQ     401$          ;YES GO ON
8924 041204 104002          ERROR    +2           ;MMU ERROR
8925                               ;NO GO TO ERROR
8926 041206 020127 000244          CMP     R1,@244      ;IS R1 CORRECT
8927 041212 001401          BEQ     402$          ;YES GO ON
8928 041214 104002          ERROR    +2           ;MMU ERROR
8929                               ;NO GO TO ERROR
8930 041216 005037 177776          CLR     @177776      ;SET PSW TO KERNEL MODE
8931 041222 022627 113672          CMP     (SP)+,@113672 ;IS STACK CORRECT
8932 041226 001401          BEQ     403$          ;YES GO ON
8933 041230 104002          ERROR    +2           ;MMU ERROR
8934                               ;NO GO TO ERROR
8935 041232 021627 177777          CMP     (SP),@177777 ;IS STACK CORRECT
8936 041236 001401          BEQ     404$          ;YES GO ON
8937 041240 104002          ERROR    +2           ;MMU ERROR
8938                               ;NO GO TO ERROR
8939 041242 005003          CLR     R3           ;SETUP SOURCE FOR NEXT TEST
8940 041244 005237 177572          INC     @177572      ;TURN MMU ON
8941 041250 106503          MFPD   R3           ;TEST INSTRUCTION
8942 041252 022737 000004 177776      CMP     @4,@177776   ;IS PSW CORRECT
8943 041260 001401          BEQ     6$           ;YES GO ON
8944 041262 104002          ERROR    +2           ;MMU ERROR
8945                               ;NO GO TO ERROR
8946 041264 005037 177572          CLR     @177572      ;TURN MMU OFF
8947 041270 020327 000000          CMP     R3,@0        ;IS R3 CORRECT
8948 041274 001401          BEQ     7$           ;YES GO ON
8949 041276 104002          ERROR    +2           ;MMU ERROR
8950                               ;NO GO TO ERROR
8951 041300 022627 000000          CMP     (SP)+,@0     ;IS STACK CORRECT
8952 041304 001401          BEQ     8$           ;YES GO ON
8953 041306 104002          ERROR    +2           ;MMU ERROR
8954                               ;NO GO TO ERROR
8955 041310 022627 177777          CMP     (SP)+,@177777 ;IS STACK CORRECT
8956 041314 001401          BEQ     9$           ;YES GO ON
8957 041316 104002          ERROR    +2           ;MMU ERROR
8958                               ;NO GO TO ERROR
8959 041320 012637 000244          MOV     (SP)+,@244   ;RESTORE TEST LOCATION
8960
8961                               ;
8964 041324          ;TSM6C:
8965                               ;
8966 041324 005037 177572          CLR     @177572      ;MMU OFF
8967 041330 005037 003030          CLR     @FLAG        ;CLEAR MMU ABORT FLAG

```


MEMORY MANAGEMENT TESTS

```

9025 041626 020127 000244      301$:  CMP      R1,#244          ;IS R1 CORRECT
9026 041632 001401              BEQ      302$          ;YES GO ON
9027 041634 104002              ERROR    +2           ;MMU ERROR
9028                               ;NO GO TO ERROR
9029 041636 005037 177776      302$:  CLR      @#177776       ;SET PSW TO KERNEL MODE
9030 041642 021627 177777      CMP      (SP),@#177777 ;IS STACK CORRECT
9031 041646 001401              BEQ      304$          ;YES GO ON
9032 041650 104002              ERROR    +2           ;MMU ERROR
9033                               ;NO GO TO ERROR
9034 041652 012737 030017 177776 304$:  MOV      @#30017,@#177776 ;SETUP PSW
9035 041660 012746 122347      MOV      @#122347,-(SP) ;SETUP TEST DATA
9036 041664 012701 000246      MOV      @#246,R1      ;SETUP R1
9037 041670 005237 177572      INC      @#177572     ;TURN MMU ON
9038 041674 006641              MTPI     -(R1)         ;TEST INSTRUCTION
9039 041676 022737 030011 177776  CMP      @#30011,@#177776 ;IS PSW CORRECT
9040 041704 0 101              BEQ      400$          ;YES GO ON
9041 041706 104002              ERROR    +2           ;MMU ERROR
9042                               ;NO GO TO ERROR
9043 041710 005037 177572      400$:  CLR      @#177572     ;TURN MMU OFF
9044 041714 023727 000244 122347  CMP      @#244,@#122347 ;IS TEST LOCATION CORRECT
9045 041722 001401              BEQ      401$          ;YES GO ON
9046 041724 104002              ERROR    +2           ;MMU ERROR
9047                               ;NO GO TO ERROR
9048 041726 020127 000244      401$:  CMP      R1,#244          ;IS R1 CORRECT
9049 041732 001401              BEQ      402$          ;YES GO ON
9050 041734 104002              ERROR    +2           ;MMU ERROR
9051                               ;NO GO TO ERROR
9052 041736 005037 177776      402$:  CLR      @#177776       ;SET PSW TO KERNEL MODE
9053 041742 021627 177777      CMP      (SP),@#177777 ;IS STACK CORRECT
9054 041746 001401              BEQ      404$          ;YES GO ON
9055 041750 104002              ERROR    +2           ;MMU ERROR
9056                               ;NO GO TO ERROR
9057 041752 005046              CLR      -(SP)         ;SETUP STACK FOR NEXT TEST
9058 041754 005237 177572      INC      @#177572     ;TURN MMU ON
9059 041760 006603              MTPI     R3            ; TEST INSTRUCTION
9060 041762 022737 000004 177776  CMP      @#4,@#177776  ;IS PSW CORRECT
9061 041770 001401              BEQ      5$           ;YES GO ON
9062 041772 104002              ERROR    +2           ;MMU ERROR
9063                               ;NO GO TO ERROR
9064 041774 005037 177572      5$:   CLR      @#177572     ;TURN MMU OFF
9065 042000 020327 000000      CMP      R3,#0         ;IS R3 CORRECT
9066 042004 001401              BEQ      6$           ;YES GO ON
9067 042006 104002              ERROR    +2           ;MMU ERROR
9068                               ;NO GO TO ERROR
9069 042010 022627 177777      6$:   CMP      (SP)+,@#177777 ;IS STACK CORRECT
9070 042014 001401              BEQ      7$           ;YES GO ON
9071 042016 104002              ERROR    +2           ;MMU ERROR
9072                               ;NO GO TO ERROR
9073 042020 012637 000244      7$:   MOV      (SP)+,@#244   ;RESTORE TEST LOCATION
9074
9075
9078 042024      ;
9079      ;
9080 042024 005037 177572      ; TEST MTPD (MOVE TO PREVIOUS DATA SPACE)
9081 042030 005037 003030      CLR      @#177572     ;MMU OFF
9082 042034 012737 140000 177776  CLR      @#FLAG       ;CLEAR MMU ABORT FLAG
9083 042042 010637 003040      MOV      @#140000,@#177776 ;POINT TO USER SPACE
          MOV      R6,@#SAVUSE ;SAVE USER SP
    
```

MEMORY MANAGEMENT TESTS

9084	042046	012737	040000	177776		MOV	#40000,#177776	;PCINT TO SUPERVISOR SPACE
9085	042054	010637	003036			MOV	R6,#SAVSUP	;SAVE SUPERVISOR SP
9086	042060	012737	030000	177776		MOV	#30000,#177776	;SETUP PSW
9087	042066	004767	075460			JSR	PC,MMU	;INIT MMU
9088	042072	012737	000027	172516		MOV	#27,#172516	;SETUP MMR3
9089	042100	013746	000244			MOV	#244,-(SP)	;SAVE DATA AT TEST LOCATION
9090	042104	012746	177777			MOV	#177777,-(SP)	;PUT KNOWN DATA ON STACK
9091	042110	012746	100004			MOV	#100004,-(SP)	;PUT TEST DATA ON STACK
9092	042114	012737	177777	000244		MOV	#177777,#244	;PUT KNOWN DATA AT TEST LOCATION
9093	042122	012767	077400	135450		MOV	#77400,UIPDRO	;SETUP UIPDRO TO ABORT
9094	042130	012703	000244			MOV	#244,R3	;SETUP POINTER TO TEST LOCATION
9095	042134	005237	177572			INC	#177572	;TURN MMU ON
9096	042140	106623				MTPD	(R3)+	; TEST INSTRUCTION
9097	042142	022737	030010	177776		CMP	#30010,#177776	;IS PSW CORRECT
9098	042150	001401				BEQ	1#	;YES GO ON
9099	042152	104002				ERROR	+2	;MMU ERROR
9100								;NO GO TO ERROR
9101	042154	005037	177572		1#:	CLR	#177572	;TURN MMU OFF
9102	042160	012737	140000	177776		MOV	#140000,#177776	;POINT TO USER SPACE
9103	042166	020637	003040			CMP	R6,#SAVUSE	;IS USER SP CORRECT
9104	042172	001401				BEQ	100#	;YES GO ON
9105	042174	104002				ERROR	+2	;MMU ERROR
9106								;NO GO TO ERROR
9107	042176	012737	040000	177776	100#:	MOV	#40000,#177776	;POINT TO SUPERVISOR SPACE
9108	042204	020637	003036			CMP	R6,#SAVSUP	;IS SUPERVISOR SP CORRECT
9109	042210	001401				BEQ	200#	;YES GO ON
9110	042212	104002				ERROR	+2	;MMU ERROR
9111								;NO GO TO ERROR
9112	042214	023727	000244	100004	200#:	CMP	#244,#100004	;IS TEST LOCATION CORRECT
9113	042222	001401				BEQ	2#	;YES GO ON
9114	042224	104002				ERROR	+2	;MMU ERROR
9115								;NO GO TO ERROR
9116	042226	020327	000246		2#:	CMP	R3,#246	;IS R3 CORRECT
9117	042232	001401				BEQ	3#	;YES GO ON
9118	042234	104002				ERROR	+2	;MMU ERROR
9119								;NO GO TO ERROR
9120	042236	005037	177776		3#:	CLR	#177776	;SET PSW TO KERNEL MODE
9121	042242	021627	177777			CMP	(SP),#177777	;IS KERNEL STACK CORRECT
9122	042246	001401				BEQ	4#	;YES GO ON
9123	042250	104002				ERROR	+2	;MMU ERROR
9124								;NO GO TO ERROR
9125	042252	012737	030017	177776	4#:	MOV	#30017,#177776	;SETUP PSW
9126	042260	012746	100737			MOV	#100737,-(SP)	;SETUP TEST DATA
9127	042264	012701	000244			MOV	#244,R1	;SETUP R1
9128	042270	005237	177572			INC	#177572	;TURN MMU ON
9129	042274	106611				MTPD	(R1)	;TEST INSTRUCTION
9130	042276	022737	030011	177776		CMP	#30011,#177776	;IS PSW CORRECT
9131	042304	001401				BEQ	300#	;YES GO ON
9132	042306	104002				ERROR	+2	;MMU ERROR
9133								;NO GO TO ERROR
9134	042310	005037	177572		300#:	CLR	#177572	;TURN MMU OFF
9135	042314	023727	000244	100737		CMP	#244,#100737	;IS TEST LOCATION CORRECT
9136	042322	001401				BEQ	301#	;YES GO ON
9137	042324	104002				ERROR	+2	;MMU ERROR
9138								;NO GO TO ERROR
9139	042326	020127	000244		301#:	CMP	R1,#244	;IS R1 CORRECT
9140	042332	001401				BEQ	302#	;YES GO ON

MEMORY MANAGEMENT TESTS

```

9141 042334 104002          ERROR +2          ;MMU ERROR
9142                                ;NO GO TO ERROR
9143 042336 005037 177776    302$: CLR 0#177776          ;SET PSW TO KERNEL MODE
9144 042342 021627 177777    CMP (SP),#177777      ;IS STACK CORRECT
9145 042346 001401          BEQ 304$             ;YES GO ON
9146 042350 104002          ERROR +2          ;MMU ERROR
9147                                ;NO GO TO ERROR
9148 042352 012737 030017 177776 304$: MOV #30017,0#177776    ;SETUP PSW
9149 042360 012746 156711    MOV #156711,-(SP)     ;SETUP TEST DATA
9150 042364 012701 000246    MOV #246,R1          ;SETUP R1
9151 042370 005237 177572    INC 0#177572        ;TURN MMU ON
9152 042374 106641          MTPD -(R1)          ;TEST INSTRUCTION
9153 042376 022737 030011 177776  CMP #30011,0#177776  ;IS PSW CORRECT
9154 042404 001401          BEQ 400$             ;YES GO ON
9155 042406 104002          ERROR +2          ;MMU ERROR
9156                                ;NO GO TO ERROR
9157 042410 005037 177572    400$: CLR 0#177572        ;TURN MMU OFF
9158 042414 023727 000244 156711  CMP 0#244,#156711    ;IS TEST LOCATION CORRECT
9159 042422 001401          BEQ 401$             ;YES GO ON
9160 042424 104002          ERROR +2          ;MMU ERROR
9161                                ;NO GO TO ERROR
9162 042426 020127 000244    401$: CMP R1,#244      ;IS R1 CORRECT
9163 042432 001401          BEQ 402$             ;YES GO ON
9164 042434 104002          ERROR +2          ;MMU ERROR
9165                                ;NO GO TO ERROR
9166 042436 005037 177776    402$: CLR 0#177776        ;SET PSW TO KERNEL MODE
9167 042442 021627 177777    CMP (SP),#177777    ;IS STACK CORRECT
9168 042446 001401          BEQ 404$             ;YES GO ON
9169 042450 104002          ERROR +2          ;MMU ERROR
9170                                ;NO GO TO ERROR
9171 042452 003046          404$: CLR -(SP)        ;SETUP STACK FOR NEXT TEST
9172 042454 005237 177572    INC 0#177572        ;TURN MMU ON
9173 042460 106603          MTPD R3            ;TEST INSTRUCTION
9174 042462 022737 000004 177776  CMP #4,0#177776     ;IS PSW CORRECT
9175 042470 001401          BEQ 5$              ;YES GO ON
9176 042472 104002          ERROR +2          ;MMU ERROR
9177                                ;NO GO TO ERROR
9178 042474 005037 177572    5$: CLR 0#177572        ;TURN MMU OFF
9179 042500 020327 000000    CMP R3,#0           ;IS R3 CORRECT
9180 042504 001401          BEQ 6$              ;YES GO ON
9181 042506 104002          ERROR +2          ;MMU ERROR
9182                                ;NO GO TO ERROR
9183 042510 022627 177777    6$: CMP (SP)+,#177777 ;IS STACK CORRECT
9184 042514 001401          BEQ 7$              ;YES GO ON
9185 042516 104002          ERROR +2          ;MMU ERROR
9186                                ;NO GO TO ERROR
9187 042520 012637 000244    7$: MOV (SP)+,0#244   ;RESTORE TEST LOCATION
9188
9189
9192 042524          ;TSMMU7:
9193          ;
9194 042524 005037 177572    CLR 0#177572        ;MMU OFF
9195 042530 005067 140274    CLR FLAG           ;CLEAR MMU ABORT FLAG
9196 042534 013746 000214    MOV 0#214,-(SP)    ;SAVE DATA AT TEST LOCATIONS
9197 042540 013746 000216    MOV 0#216,-(SP)
9198 042544 005067 140272    CLR SAVMRO         ;CLEAR STATUS REGS SAVE AREAS
9199 042550 005067 140270    CLR SAVMR1

```

MEMORY MANAGEMENT TESTS

```

9200 042554 005067 140266      CLR      SAVMR2      ;
9201 042560 004767 074766      JSR      PC,MMU      ;INIT MMU
9202 042564 012737 030000 177776  MOV      #30000,#177776 ;SETUP PSW
9203 042572 012702 000200      MOV      #200,R2      ;
9204 042576 012737 077400 177600  MOV      #77400,#177600 ;SETUP FOR AN ABORT
9205 042604 004767 000164      JSR      PC,TS7      ;CAUSE AN ABORT TO OCCUR AND
9206                                     ;THEN CHECK IF ABORT FLAG REGISTERED
9207                                     ;THIS EVENT AND CHECK IF STATUS REGS
9208                                     ;CONTAINED EXPECTED VALUES.
9209                                     ;IF NO ABORT OCCURRED THEN GO TO ERROR
9210                                     ;OTHERWISE CONTINUE.
9211 042610 012737 077404 177600  MOV      #77404,#177600 ;SETUP FOR AN ABORT
9212 042616 004767 000152      JSR      PC,TS7      ;CAUSE AN ABORT TO OCCUR AND
9213                                     ;THEN CHECK IF ABORT FLAG REGISTERED
9214                                     ;THIS EVENT AND CHECK IF STATUS REGS
9215                                     ;CONTAINED EXPECTED VALUES.
9216                                     ;IF NO ABORT OCCURRED THEN GO TO ERROR
9217                                     ;OTHERWISE CONTINUE.
9218 042622 012701 000220      MOV      #220,R1      ;
9219 042626 004767 074720      JSR      PC,MMU      ;INIT MMU
9220 042632 005003      CLR      R3          ;SETUP MMR1 EXPECTED DATA
9221 042634 012767 000001 140166  MOV      #1,FLAG      ;SETUP FLAG FOR AN ABORT
9222 042642 012737 000001 177572  MOV      #1,#177572    ;TURN MMU ON
9223 042650 012737 100000 177776  MOV      #100000,#177776 ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
9224 042656 012241      MOV      (R2)+, -(R1) ;CAUSE AN ABORT
9225 042660 004767 000220      JSR      PC,TSM7     ;CHECK IF AN ABORT OCCURRED BY
9226                                     ;CHECKING ABORT FLAG AND STATUS REGS
9227                                     ;IF NO ABORT OCCURRED THEN GO TO ERROR
9228                                     ;OTHERWISE CONTINUE.
9229 042664 005067 140152      CLR      SAVMR0     ;CLEAR STATUS REGS SAVE AREAS
9230 042670 005067 140150      CLR      SAVMR1     ;
9231 042674 005067 140146      CLR      SAVMR2     ;
9232 042700 012703 000022      MOV      #22,R3      ;SETUP MMR1 EXPECTED DATA
9233 042704 012767 000001 140116  MOV      #1,FLAG      ;SETUP FLAG FOR AN ABORT
9234 042712 012737 000001 177572  MOV      #1,#177572    ;TURN MMU ON
9235 042720 012737 020000 177776  MOV      #20000,#177776 ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
9236 042726 006522      MFPI      (R2)+      ;CAUSE AN ABORT
9237 042730 004767 000150      JSR      PC,TSM7     ;CHECK IF AN ABORT OCCURRED BY
9238                                     ;CHECKING ABORT FLAG AND STATUS REGS
9239                                     ;IF NO ABORT OCCURRED THEN GO TO ERROR
9240                                     ;OTHERWISE CONTINUE.
9241 042734 012737 030000 177776  MOV      #30000,#177776 ;SETUP PSW
9242 042742 012737 077400 177600  MOV      #77400,#177600 ;SETUP FOR AN ABORT
9243 042750 005037 177572      CLR      #177572     ;MMU OFF
9244 042754 006522      MFPI      (R2)+      ;TRY TO CAUSE AN ABORT
9245 042756 012603      MOV      (SP)+,R3     ;POP THE STACK
9246 042760 012637 000216      MOV      (SP)+,#216   ;RESTORE DATA AT TEST LOCATIONS
9247 042764 012637 000214      MOV      (SP)+,#214   ;
9248                                     ;
9249 042770 000167 000154      JMP      TS7FIN      ;
9250                                     ;
9251                                     ;ROUTINE TO CAUSE AND CHECK NONRESIDENT ABORTS
9252                                     ;
9253 042774 012767 000001 140026  TS7:  MOV      #1,FLAG      ;SETUP FOR AN ABORT
9254 043002 012737 000001 177572  MOV      #1,#177572    ;TURN MMU ON
9255 043010 010701      MOV      R7,R1      ;SAVE PC
9256 043012 006522      MFPI      (R2)+      ;CAUSE AN ABORT

```

MEMORY MANAGEMENT TESTS

```

9257 043014 022767 000000 140006      CMP      #0,FLAG      ;DID AN ABORT OCCUR
9258 043022 001401      BEQ      OK7          ;IF YES GO ON
9259 043024 104002      ERROR   +2          ;MMU ERROR
9260                                ;IF NO GO TO ERROR
9261 043026 105067 140010      OK7:    CLRB     SAVMRO      ;SETUP EXPECTED DATA
9262 043032 022767 100000 140002      CMP      #100000,SAVMRO ; TEST MMRO FOR EXPECTED VALUE
9263 043040 001401      BEQ      OKA7        ;IF OK THEN CONTINUE
9264 043042 104002      ERROR   +2          ;MMU ERROR
9265                                ;NOT OK THEN GO TO ERROR
9266 043044 026727 137774 000022      OKA7:   CMP      SAVMR1,#22 ; TEST MMR1 FOR EXPECTED VALUE
9267 043052 001401      BEQ      OKAY7       ;IF OK THEN CONTINUE
9268 043054 104002      ERROR   +2          ;MMU ERROR
9269                                ;NOT OK THEN GO TO ERROR
9270 043056 026701 137764      OKAY7:  CMP      SAVMR2,R1   ; TEST MMR2 FOR EXPECTED VALUE
9271 043062 001401      BEQ      OKAY7A      ;IF OK THEN CONTINUE
9272 043064 104002      ERROR   +2          ;MMU ERROR
9273                                ;NOT OK THEN GO TO ERROR
9274 043066 005067 137750      OKAY7A: CLR     SAVMRO      ;CLEAR STATUS REGS SAVE AREAS
9275 043072 005067 137746      CLR     SAVMR1
9276 043076 005067 137744      CLR     SAVMR2
9277 043102 000207      RTS     PC          ;RETURN
9278
9279      ;ROUTINE TO CHECK IF A NONRESIDENT ABORT OCCURRED
9280
9281 043104 022767 000000 137716      TSM7:   CMP      #0,FLAG      ;DID AN ABORT OCCUR
9282 043112 001401      BEQ      TSMA        ;IF YES GO ON
9283 043114 104002      ERROR   +2          ;MMU ERROR
9284                                ;IF NO THEN GO TO ERROR
9285 043116 042737 040377 003042      TSMA:   BIC     #40377,#SAVMRO ;SETUP EXPECTED DATA
9286 043124 022767 100000 137710      CMP      #100000,SAVMRO ; TEST MMRO FOR EXPECTED VALUE
9287 043132 001401      BEQ      TSMB        ;IF OK THEN CONTINUE
9288 043134 104002      ERROR   +2          ;MMU ERROR
9289                                ;IF NO THEN GO TO ERROR
9290 043136 020367 137702      TSMB:   CMP      R3,SAVMR1   ; TEST MMR1 FOR EXPECTED VALUE
9291 043142 001401      BEQ      TSMC        ;IF OK THEN CONTINUE
9292 043144 104002      ERROR   +2          ;MMU ERROR
9293                                ;IF NOT OK THEN GO TO ERROR
9294 043146 000207      TSMC:   RTS     PC          ;RETURN
9295
9296 043150 000240      ;TS7FIN: NOP
9299 043152      ;TSMU8:
9300      ;
9301 043152 005037 177572      ;
9302 043156 005067 137646      CLR     #177572      ;MMU OFF
9303 043162 013746 000244      CLR     FLAG         ;CLEAR MMU ABORT FLAG
9304 043166 013746 000246      MOV     #244,-(SP)   ;SAVE DATA AT TEST LOCATIONS
9305 043172 005067 137644      MOV     #246,-(SP)
9306 043176 005067 137642      CLR     SAVMRO      ;
9307 043202 005067 137640      CLR     SAVMR1      ;CLEAR STATUS REGS SAVE AREAS
9308 043206 004767 074340      CLR     SAVMR2
9309 043212 012737 030000 177776      JSR     PC,MMU      ;
9310 043220 012702 000244      MOV     #30000,#177776 ;INIT MMU
9311 043224 012737 077402 177600      MOV     #244,R2      ;SETUP PSW
9312 043232 012746 000246      MOV     #77402,#177600 ;
9313 043236 012767 000001 137564      MOV     #246,-(SP)   ;SETUP FOR AN ABORT
9314 043244 012737 000001 177572      MOV     #1,FLAG     ;PUSH DATA ONTO THE STACK
9315 043252 010701      MOV     #1,#177572  ;SETUP FLAG FOR AN ABORT
                                MOV     R7,R1      ;TURN MMU ON
                                ;SAVE PC

```

MEMORY MANAGEMENT TESTS

```

9316 043254 006622          MTPI      (R2)+          ;CAUSE ABORT
9317 043256 022767 000000 137544  CMP      #0,FLAG      ;DID ABORT OCCUR
9318 043264 001401          BEQ      1#          ;IF YES THEN GO ON
9319 043266 104002          ERROR    +2          ;MMU ERROR
9320                                ;IF NO THEN GO TO ERROR
9321 043270 105067 137546 1# :  CLR      SAVMRO      ;SETUP EXPECTED DATA
9322 043274 022767 020000 137540  CMP      #20000,SAVMRO ; TEST MMRO FOR EXPECTED VALUE
9323 043302 001401          BEQ      2#          ;IF OK THEN CONTINUE
9324 043304 104002          ERROR    +          ;MMU ERROR
9325                                ;OTHERWISE GO TO ERROR
9326 043306 022767 011026 137530 2# :  CMP      #11026,SAVMR1 ; TEST MMR1 FOR EXPECTED VALUE
9327 043314 001401          BEQ      3#          ;IF OK THEN CONTINUE
9328 043316 104002          ERROR    +          ;MMU ERROR
9329                                ;OTHERWISE GO TO ERROR
9330 043320 020167 137522 3# :  CMP      R1,SAVMR2    ; TEST MMR2 FOR EXPECTED VALUE
9331 043324 001401          BEQ      4#          ;IF OK THEN CONTINUE
9332 043326 104002          ERROR    +2          ;MMU ERROR
9333                                ;OTHERWISE GO TO ERROR
9334 043330 012737 030000 177776 4# :  MOV      #30000,#177776 ;SETUP PSW
9335 043336 012746 000002          MOV      #2,-(SP)     ;PUSH DATA ONTO STACK
9336 043342 006622          MTPI      (R2)+          ;TRY TO CAUSE ABORT
9337 043344 012637 000246          MOV      (SP)+,#246   ;RESTORE DATA AT TEST LOCATIONS
9338 043350 012637 000244          MOV      (SP)+,#244   ;
9339
9342 043354          TSMMU9:
9343          ; TEST PAGE LENGTH ERROR ABORTS
9344 043354 005037 177572          CLR      #177572     ;MMU OFF
9345 043360 005067 137444          CLR      FLAG        ;CLEAR ABORT FLAG
9346 043364 005067 137452          CLR      SAVMRO      ;CLEAR STATUS REGS SAVE AREAS
9347 043370 005067 137450          CLR      SAVMR1      ;
9348 043374 005067 137446          CLR      SAVMR2      ;
9349 043400 012737 030000 177776  MOV      #30000,#177776 ;SETUP PSW
9350 043406 004767 074140          JSR      PC,MMU      ;INIT MMU
9351 043412 012703 043674          MOV      #PLF0,R3    ;LET R3, R1, AND R2 POINT TO THE
9352 043416 012701 043746          MOV      #BN0,R1     ;UPWARD EXPANSION TABLES
9353 043422 012702 044016          MOV      #ABORT0,R2  ;
9354 043426 012737 000026 172516  MOV      #26,#172516 ;DISABLE USER DATA SPACE
9355 043434 004767 000050          JSR      PC,TSM9     ;TURN MMU ON
9356                                ;DO RELOCATIONS FOR THE DIFFERENT
9357                                ;VALUES OF THE PAGE LENGTH FIELD AND
9358                                ;BLOCK NUMBER. IF AN ABORT OCCURS
9359                                ;CHECK TO SEE IF IT WAS SUPPOSED TO.
9360                                ;AND IF YES CHECK ABORT FLAG AND
9361                                ;STATUS REGISTERS.
9362 043440 012703 044066          MOV      #PLF1,R3    ;LET R3, R1, AND R2 POINT TO THE
9363 043444 012701 044136          MOV      #BN1,R1     ;DOWNWARD EXPANSION TABLES
9364 043450 012702 044204          MOV      #ABORT7,R2  ;
9365 043454 004767 000030          JSR      PC,TSM9     ;TURN MMU ON
9366                                ;DO RELOCATIONS FOR THE DIFFERENT
9367                                ;VALUES OF THE PAGE LENGTH FIELD AND
9368                                ;BLOCK NUMBER. IF AN ABORT OCCURS
9369                                ;CHECK TO SEE IF IT WAS SUPPOSED TO.
9370                                ;AND IF YES CHECK ABORT FLAG AND
9371                                ;STATUS REGISTERS.
9372 043460 005037 177572          CLR      #177572     ;MMU OFF
9373 043464 012703 044076          MOV      #PLF1+10,R3 ;POINT TO A VALUE WHICH SHOULD CAUSE
9374 043470 012701 044146          MOV      #BN1+10,R1 ;AN ABORT IF MMU IS ON.

```

MEMORY MANAGEMENT TESTS

```

9375 043474 011337 177600      MOV      (R3),0#177600      ;SETUP UIPDRO
9376 043500 006521              MFPI     (R1)+              ;DO A RELOCATION
9377 043502 012605              MOV      (SP)+,R5          ;POP THE STACK
9378
9379 043504 000167 000542      JMP      TS9FIN
9380
9381      ;ROUTINE TO CAUSE AND CHECK PAGE LENGTH ERROR ABORTS
9382      ;
9383 043510 012337 177600      TSM9:   MOV      (R3)+,0#177600  ;SETUP UIPDRO
9384 043514 010100              MOV      R1,R0             ;SAVE A COPY OF R1
9385 043516 012767 000001 137304  MOV      #1,FLAG          ;SETUP FOR AN ABORT
9386 043524 012737 000001 177572  MOV      #1,0#177572      ;TURN MMU ON
9387 043532 010704              MOV      R7,R4             ;SAVE PC
9388 043534 006530              MFPI     0(R0)+           ;DO A RELOCATION OPERATION
9389 043536 021227 000000      CMP      (R2),#0          ;WAS AN ABORT SUPPOSED TO OCCUR
9390 043542 001007              BNE     2#                ;IF YES GO TO 2#
9391 043544 012605              MOV      (SP)+,R5          ;POP THE STACK
9392 043546 022767 000001 137254  CMP      #1,FLAG          ;DID AN ABORT OCCUR
9393 043554 001401              BEQ     1#                ;NO GO ON
9394 043556 104002              ERROR   +2                ;MMU ERROR
9395
9396 043560 000425              1#:    BR      6#                ;YES GO TO ERROR
9397 043562 022767 000000 137240  2#:    CMP      #0,FLAG          ;DID AN ABORT OCCUR
9398 043570 001401              BEQ     3#                ;YES GO ON
9399 043572 104002              ERROR   +2                ;MMU ERROR
9400
9401 043574 105067 137242      3#:    CLRB   SAVMRO           ;SETUP EXPECTED DATA
9402 043600 022767 040000 137234  CMP      #40000,SAVMRO    ; TEST MMRO FOR EXPECTED VALUE
9403 043606 001401              BEQ     4#                ;IF OK THEN CONTINUE
9404 043610 104002              ERROR   +2                ;MMU ERROR
9405
9406 043612 022767 000020 137224  4#:    CMP      #20,SAVMR1      ; TEST MMR1 FOR EXPECTED VALUE
9407 043620 001401              BEQ     5#                ;IF OK THEN CONTINUE
9408 043622 104002              ERROR   +2                ;MMU ERROR
9409
9410 043624 020467 137216      5#:    CMP      R4,SAVMR2      ; TEST MMR2 FOR EXPECTED VALUE
9411 043630 001401              BEQ     6#                ;IF OK THEN CONTINUE
9412 043632 104002              ERROR   +2                ;MMU ERROR
9413
9414 043634 005067 137170      6#:    CLR     FLAG            ;CLEAR MMU ABORT FLAG
9415 043640 005067 137176      CLR     SAVMRO           ;CLEAR STATUS REGS SAVE AREAS
9416 043644 005067 137174      CLR     SAVMR1           ;
9417 043650 005067 137172      CLR     SAVMR2           ;
9418 043654 005201              INC     R1                ;POINT TO NEXT ENTRY
9419 043656 005201              INC     R1                ;
9420 043660 005202              INC     R2                ;
9421 043662 005202              INC     R2                ;
9422 043664 021327 000777      CMP      (R3),#777        ;HAVE ALL ENTRIES BEEN TRIED
9423 043670 001307              BNE     TSM9              ;NO REPEAT
9424 043672 000207              RTS     PC                 ;YES RETURN
9425
9426      ;UPWARD EXPANSION TABLES
9427      ;
9428 043674 070006      PLFO:   .WORD   70006
9429 043676 070006      .WORD   70006
9430 043700 070006      .WORD   70006
9431 043702 013406      .WORD   13406

```

MEMORY MANAGEMENT TESTS

9432	043704	020006	.WORD	20006
9433	043706	004006	.WORD	04006
9434	043710	040006	.WORD	40006
9435	043712	070006	.WORD	70006
9436	043714	024006	.WORD	24006
9437	043716	004006	.WORD	04006
9438	043720	014006	.WORD	14006
9439	043722	012006	.WORD	12006
9440	043724	002006	.WORD	02006
9441	043726	001406	.WORD	01406
9442	043730	004006	.WORD	04006
9443	043732	002006	.WORD	02006
9444	043734	000406	.WORD	00406
9445	043736	007406	.WORD	07406
9446	043740	001006	.WORD	01006
9447	043742	003406	.WORD	03406
9448	043744	000777	.WORD	777
9449	043746	013000	BNO: .WORD	013000
9450	043750	016000	.WORD	016000
9451	043752	017000	.WORD	017000
9452	043754	002700	.WORD	002700
9453	043756	014000	.WORD	014000
9454	043760	002000	.WORD	002000
9455	043762	004000	.WORD	004000
9456	043764	007000	.WORD	007000
9457	043766	002000	.WORD	002000
9458	043770	000700	.WORD	000700
9459	043772	004000	.WORD	004000
9460	043774	001000	.WORD	001000
9461	043776	000300	.WORD	000300
9462	044000	000400	.WORD	000400
9463	044002	001400	.WORD	001400
9464	044004	000600	.WORD	000600
9465	044006	000200	.WORD	000200
9466	044010	001700	.WORD	001700
9467	044012	000300	.WORD	000300
9468	044014	000700	.WORD	000700
9469	044016	000000	ABORTO: .WORD	0
9470	044020	000000	.WORD	0
9471	044022	000001	.WORD	1
9472	044024	000000	.WORD	0
9473	044026	000001	.WORD	1
9474	044030	000001	.WORD	1
9475	044032	000000	.WORD	0
9476	044034	000000	.WORD	0
9477	044036	000000	.WORD	0
9478	044040	000000	.WORD	0
9479	044042	000001	.WORD	1
9480	044044	000000	.WORD	0
9481	044046	000000	.WORD	0
9482	044050	000001	.WORD	1
9483	044052	000001	.WORD	1
9484	044054	000001	.WORD	1
9485	044056	000001	.WORD	1
9486	044060	000000	.WORD	0
9487	044062	000001	.WORD	1
9488	044064	000000	.WORD	0

MEMORY MANAGEMENT TESTS

9489			
9490			;DOWNWARD EXPANSION TABLES
9491			;
9492	044066	000416	PLF1: .WORD 00416
9493	044070	020016	.WORD 20016
9494	044072	024016	.WORD 24016
9495	044074	034016	.WORD 34016
9496	044076	074016	.WORD 74016
9497	044100	040016	.WORD 40016
9498	044102	020016	.WORD 20016
9499	044104	000016	.WORD 00016
9500	044106	030016	.WORD 30016
9501	044110	010016	.WORD 10016
9502	044112	014016	.WORD 14016
9503	044114	004016	.WORD 04016
9504	044116	002016	.WORD 02016
9505	044120	000416	.WORD 00416
9506	044122	000016	.WORD 00016
9507	044124	003416	.WORD 03416
9508	044126	001016	.WORD 01016
9509	044130	001416	.WORD 01416
9510	044132	000416	.WORD 00416
9511	044134	000777	.WORD 777
9512	044136	000100	BN1: .WORD 000100
9513	044140	010000	.WORD 010000
9514	044142	006000	.WORD 006000
9515	044144	016000	.WORD 016000
9516	044146	016000	.WORD 016000
9517	044150	004000	.WORD 004000
9518	044152	000000	.WORD 000000
9519	044154	000000	.WORD 000000
9520	044156	004000	.WORD 004000
9521	044160	004000	.WORD 004000
9522	044162	004000	.WORD 004000
9523	044164	000000	.WORD 000000
9524	044166	000300	.WORD 000300
9525	044170	000000	.WORD 000000
9526	044172	000400	.WORD 000400
9527	044174	001000	.WORD 001000
9528	044176	000100	.WORD 000100
9529	044200	000400	.WORD 000400
9530	044202	000200	.WORD 000200
9531	044204	000000	ABORT7: .WORD 0
9532	044206	000000	.WORD 0
9533	044210	000000	.WORD 0
9534	044212	000000	.WORD 0
9535	044214	000001	.WORD 1
9536	044216	000001	.WORD 1
9537	044220	000001	.WORD 1
9538	044222	000000	.WORD 0
9539	044224	000001	.WORD 1
9540	044226	000000	.WORD 0
9541	044230	000000	.WORD 0
9542	044232	000001	.WORD 1
9543	044234	000001	.WORD 1
9544	044236	000001	.WORD 1
9545	044240	000000	.WORD 0

MEMORY MANAGEMENT TESTS

```

9546 044242 000000 .WORD 0
9547 044244 000001 .WORD 1
9548 044246 000000 .WORD 0
9549 044250 000000 .WORD 0
9550
9551 044252 000240 ; TS9FIN: NOP
9554 044254 ; TSMM10:
9555 ;
; FUNCTIONAL TEST OF BITS <6:1> OF MMRO
9556 044254 005037 177572 CLR #0177572 ;MMU OFF
9557 044260 005067 136544 CLR FLAG ;CLEAR MMU ABORT FLAG
9558 044264 005067 136552 CLR SAVMRO ;CLEAR STATUS REGS SAVE AREAS
9559 044270 005067 136550 CLR SAVMR1 ;
9560 044274 005067 136546 CLR SAVMR2 ;
9561 044300 004767 073246 JSR PC,MMU ;INIT MMU
9562 044304 005037 177776 CLR #0177776 ;INIT PSW: PREVIOUS MODE = KERNAL
9563 044310 012702 020200 MOV #020200,R2 ;
9564 044314 012737 077400 172302 MOV #77400,#0172302 ;SETUP KIPDR1 TO ABORT
9565 044322 012767 000001 136500 MOV #1,FLAG ;SETUP FLAG FOR AN ABORT
9566 044330 012737 000001 177572 MOV #1,#0177572 ;TURN MMU ON
9567 044336 010701 MOV R7,R1 ;SAVE PC
9568 044340 006522 MFPI (R2)+ ;DO A RELOCATION VIA KIPAR1
9569 044342 012704 100003 MOV #100003,R4 ;SETUP EXPECTED DATA
9570 044346 004767 000202 JSR PC,TS10 ;CHECK IF AN ABORT OCCURRED AND
9571 ;IF YES CHECK BITS <6:1> OF MMRO.
9572 044352 012737 030000 177776 MOV #30000,#0177776 ;INIT PSW: PREVIOUS MODE = USER
9573 044360 004767 073166 JSR PC,MMU ;INIT MMU
9574 044364 012737 077400 177636 MOV #77400,#0177636 ;SETUP UDPR7 TO ABORT
9575 044372 012702 160000 MOV #160000,R2 ;
9576 044376 012767 000001 136424 MOV #1,FLAG ;SETUP FLAG FOR AN ABORT
9577 044404 012737 000001 177572 MOV #1,#0177572 ;TURN MMU ON
9578 044412 010701 MOV R7,R1 ;SAVE PC
9579 044414 106522 MFPD (R2)+ ;DO A RELOCATION VIA UDPAR7
9580 044416 012704 100177 MOV #100177,R4 ;SETUP EXPECTED DATA
9581 044422 004767 000126 JSR PC,TS10 ;CHECK IF AN ABORT OCCURRED AND
9582 ;IF YES CHECK BITS <6:1> OF MMRO.
9583 044426 012737 010000 177776 MOV #10000,#0177776 ;INIT PSW: PREVIOUS MODE= SUPERVISOR
9584 044434 004767 073112 JSR PC,MMU ;INIT MMU
9585 044440 012737 077400 172212 MOV #77400,#0172212 ;SETUP SIPDR5 TO ABORT
9586 044446 012702 120000 MOV #120000,R2 ;ACCESS PAGE 05
9587 044452 012767 000001 136350 MOV #1,FLAG ;SETUP FLAG FOR AN ABORT
9588 044460 012737 000001 177572 MOV #1,#0177572 ;TURN MMU ON
9589 044466 010701 MOV R7,R1 ;SAVE PC
9590 044470 006522 MFPI (R2)+ ;DO A RELOCATION VIA SIPAR5
9591 044472 012704 100053 MOV #100053,R4 ;SETUP EXPECTED DATA:ABORT, PAGE 5
9592 044476 004767 000052 JSR PC,TS10 ;CHECK IF AN ABORT OCCURRED AND
9593 ;IF YES CHECK BITS <6:1> OF MMR
9594
9595 ; TEST THAT ILLEGAL MODE CAUSES MMU ABORT
9596 ;
9597 044502 012737 020000 177776 MOV #20000,#0177776 ;INIT PSW:SET ILLEGAL PREVIOUS MODE
9598 044510 004767 073036 JSR PC,MMU ;INIT MMU
9599 044514 012702 040000 MOV #40000,R2 ;SET UP ACCESS TO PAGE 2
9600 044520 012767 000001 136302 MOV #1,FLAG ;SETUP FLAG FOR AN ABORT
9601 044526 012737 000001 177572 MOV #1,#0177572 ;TURN MMU ON
9602 044534 010701 MOV R7,R1 ;SAVE PC
9603 044536 106522 MFPD (R2)+ ;DO A RELOCATION
9604 044540 012704 100105 MOV #100105,R4 ;SETUP EXPECTED DATA:ABORT, ILLEGAL

```


MEMORY MANAGEMENT TESTS

```

9605
9606 044544 004767 000004          JSR    PC,TS10          ; PROCESSOR MODE, PAGE 02
9607                                     ;CHECK IF AN ABORT OCCURRED AND
9608                                     ;IF YES CHECK BITS <6:1> OF MMRO.
9609 044550 000167 000062          JMP    T10FIN
9610
9611          ;ROUTINE TO CHECK IF A MMU ABORT OCCURRED AND IF STATUS REG MMRO
9612          ;CONTAINS EXPECTED DATA
9613
9614 044554 022767 000000 136246  TS10:  CMP    #0,FLAG          ;DID AN ABORT OCCUR
9615 044562 001401                BEQ    1$              ;YES GO ON
9616 044564 104002                ERROR  +2              ;MMU ERROR
9617                                     ;NO GO TO ERROR
9618 044566 020467 136250  1$:    CMP    R4,SAVMRO        ; TEST MMRO FOR EXPECTED DATA
9619 044572 001401                BEQ    2$              ;OK GO ON
9620 044574 104002                ERROR  +2              ;MMU ERROR
9621                                     ;NO GO TO ERROR
9622 044576 022767 000022 136240  2$:    CMP    #22,SAVMR1      ; TEST MMR1 FOR EXPECTED DATA
9623 044604 001401                BEQ    3$              ;OK GO ON
9624 044606 104002                ERROR  +2              ;MMU ERROR
9625                                     ;NO GO TO ERROR
9626 044610 020167 136232  3$:    CMP    R1,SAVMR2      ; TEST MMR2 FOR EXPECTED DATA
9627 044614 001401                BEQ    4$              ;OK GO ON
9628 044616 104002                ERROR  +2              ;MMU ERROR
9629                                     ;NO GO TO ERROR
9630 044620 005067 136216  4$:    CLR    SAVMRO          ;CLEAR MMU STATUS REGS SAVE AREAS
9631 044624 005067 136214          CLR    SAVMR1          ;
9632 044630 005067 136212          CLR    SAVMR2          ;
9633 044634 000207                RTS     PC              ;RETURN
9634
9635 044636          ;
9638 044636          ;T10FIN:
9639          ;TSMM11:
9640 044636 005037 177572          ;
9641 044642 005067 136162          ;TEST DATA SPACE BITS MMR3
9642 044646 012737 030000 177776  CLR    #0177572        ;MMU OFF
9643 044654 012701 000026          CLR    FLAG            ;CLEAR MMU ABORT FLAG
9644 044660 012703 177610          MOV    #30000,#0177776 ;SETUP PSW
9645 044664 012704 000021          MOV    #26,R1          ;SETUP FIRST MMR3 VALUE
9646 044670 004767 000060          MOV    #177610,R3      ;POINT TO UIPDR4
9647 044674 012737 000000 177776  JSR    PC,TS11         ;SETUP SECOND MMR3 VALUE
9648 044702 012701 000023          MOV    #0,#0177776    ; TEST ENABLE USER DATA SPACE BIT
9649 044706 012703 172310          MOV    #23,R1          ;SETUP PSW
9650 044712 012704 000024          MOV    #172310,R3      ;SETUP FIRST MMR3 VALUE
9651 044716 004767 000032          MOV    #24,R4          ;POINT TO KIPDR4
9652 044722 012737 010000 177776  JSR    PC,TS11         ;SETUP SECOND MMR3 VALUE
9653 044730 012701 000025          MOV    #10000,#0177776 ; TEST ENABLE KERNEL DATA SPACE BIT
9654 044734 012703 172210          MOV    #25,R1          ;SETUP PSW
9655 044740 012704 000022          MOV    #172210,R3      ;SETUP FIRST MMR3 VALUE
9656 044744 004767 000004          JSR    PC,TS11         ;POINT TO SIPDR4
9657                                     ;SETUP SECOND MMR3 VALUE
9658 044750 000167 000120          JMP    T11FIN          ; TEST ENABLE SUPERVISOR DATA SPACE BIT
9659
9660          ;ROUTINE TO TEST ENABLE DATA SPACE BITS OF MMR3
9661          ;
9662 044754 004767 072572          ;TS11: JSR    PC,MMU          ;INIT MMU
9663 044760 010137 172516          MOV    R1,#0172516    ;DISABLE DATA SPACE OF MODE UNDER TEST

```


MEMORY MANAGEMENT TESTS

```

9723 045270 012703 006411      MOV    #6411,R3      ;SETUP EXPECTED DATA FOR MMR1
9724 045274 012767 000001 135526  MOV    #1,FLAG      ;SETUP FLAG FOR AN ABORT
9725 045302 012737 000001 177572  MOV    #1,#177572   ;TURN MMU ON
Z 9726 045310 010767 135476      MOV    R7,SLOC00    ;SAVE PC
9727 045314 112125      MOVVB  (R1)+,(R5)+   ;DO A RELOCATION
9728 045316 004767 000070      JSR    PC,TS12      ;CHECK IF AN ABORT OCCURRED AND IF
9729                                     ;YES IF MMR1 EQUALS EXPECTED DATA
9730 045322 012703 171025      MOV    #171025,R3   ;SETUP EXPECTED DATA FOR MMR1
9731 045326 012767 000001 135474  MOV    #1,FLAG      ;SETUP FLAG FOR AN ABORT
9732 045334 012737 000001 177572  MOV    #1,#177572   ;TURN MMU ON
Z 9733 045342 010767 135444      MOV    R7,SLOC00    ;SAVE PC
9734 045346 012542      MOV    (R5)+,-(R2)  ;DO A RELOCATION
9735 045350 004767 000036      JSR    PC,TS12      ;CHECK IF AN ABORT OCCURRED AND IF
9736                                     ;YES IF MMR1 EQUALS EXPECTED DATA
9737 045354 012703 012762      MOV    #12762,R3   ;SETUP EXPECTED DATA FOR MMR1
9738 045360 012767 000001 135442  MOV    #1,FLAG      ;SETUP FLAG FOR AN ABORT
9739 045366 012737 000001 177572  MOV    #1,#177572   ;TURN MMU ON
Z 9740 045374 010767 135412      MOV    R7,SLOC00    ;SAVE PC
9741 045400 014225      MOV    -(R2),(R5)+  ;DO A RELOCATION
9742 045402 004767 000004      JSR    PC,TS12      ;CHECK IF AN ABORT OCCURRED AND IF
9743                                     ;YES IF MMR1 EQUALS EXPECTED DATA
9744
9745 045406 000167 000046      JMP    T12FIN
9746
9747 ;ROUTINE TO CHECK IF AN ABORT OCCURRED AND IF MMR1 EQUALS EXPECTED DATA
9748 ;
9749 045412 022767 000000 135410  TS12:  CMP    #0,FLAG      ;DID AN ABORT OCCUR
9750 045420 001401          BEQ    1$           ;YES GO ON
9751 045422 104002          ERROR  +2          ;MMU ERROR
9752                                     ;NO GO TO ERROR
9753 045424 020367 135414      1$:   CMP    R3,SAVMR1   ;TEST MMR1 FOR EXPECTED DATA
9754 045430 001401          BEQ    2$           ;OK GO ON
9755 045432 104002          ERROR  +2          ;MMU ERROR
9756                                     ;NO GO TO ERROR
9757 045434 026767 135352 135404  2$:   CMP    SLOC00,SAVMR2 ;TEST MMR2 FOR EXPECTED DATA
9758 045442 001401          BEQ    3$           ;OK GO ON
9759 045444 104002          ERROR  +2          ;MMU ERROR
9760                                     ;NO GO TO ERROR
9761 045446 005067 135372      3$:   CLR    SAVMR1      ;CLEAR STATUS REG SAVE AREA
9762 045452 005067 135370      CLR    SAVMR2
9763 045456 000207          RTS    PC           ;RETURN
9764
9765 045460 000240      T12FIN: NOP
9775 045462      TSMM13:
9776 ;
9777 ; ADDER RELOCATION TEST PART A
;*****
;(NEED 16 BITS OF MEMORY ADDRESSING)
9778 045462 005037 177572      CLR    #177572      ;MMU OFF
9779 045466 005067 135336      CLR    FLAG         ;CLEAR MMU ABORT FLAG
9780 045472 005037 177776      CLR    #177776     ;INIT PSW
9781 045476 004767 072050      JSR    PC,MMU       ;INIT MMU
9782 045502 012737 000020 172516  MOV    #20,#172516  ;INIT MMR3
9783 045510 012703 045664      MOV    #PARAD1,R3   ;SETUP PARS WITH TEST VALUES
9784 045514 012701 045716      MOV    #PARVA1,R1
9785 045520 012133      1$:   MOV    (R1)+,#(R3)+ ;
9786 045522 021127 000333      CMP    (R1),#333   ;
9787 045526 001374          BNE    1$          ;

```

MEMORY MANAGEMENT TESTS

```

9788 045530 012703 046002      MOV      #PHY1,R3      ;SET POINTERS TO ADDER PART A
9789 045534 012701 045750      MOV      #VIR1,R1      ; TEST TABLES.
9790 045540 012702 046034      MOV      #MODE1,R2      ;
9791 045544 012237 177776      2$:     MOV      (R2)+,#0177776 ;INIT PSW
9792 045550 013305              MOV      @R3)+,R5      ;SAVE DATA AT PHYSICAL ADDRESS
9793 045552 012737 000001 177572 MOV      #1,#0177572   ;TURN MMU ON
9794 045560 006531              MFPI     @R1)+         ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
9795 045562 012604              MOV      (SP)+,R4      ;
9796 045564 005037 177572      CLR      @0177572      ;TURN MMU OFF
9797 045570 020504              CMP      R5,R4         ;IS DATA EQUAL TO EXPECTED
9798 045572 001401              BEQ      3$            ;YES GO ON
9799 045574 104002              ERROR    +2            ;MMU ERROR
9800
9801 045576 021327 000111      3$:     CMP      (R3),#111   ;NO IT IS AN ADDER ERROR
9802 045602 001360              BNE      2$            ;ARE WE READY TO TEST DATA SPACE
9803 045604 005203              INC      R3            ;NO GO TO 2$
9804 045606 005203              INC      R3            ;POINT TO DATA SPACE VALUES
9805 045610 005201              INC      R1            ;
9806 045612 005201              INC      R1            ;
9807 045614 005202              INC      R2            ;
9808 045616 005202              INC      R2            ;
9809 045620 012237 177776      MOV      (R2)+,#0177776 ;INIT PSW
9810 045624 013305              MOV      @R3)+,R5      ;SAVE DATA AT PHYSICAL ADDRESS
9811 045626 012737 000027 172516 MOV      #27,#0172516  ;INIT MMR3
9812 045634 012737 000001 177572 MOV      #1,#0177572   ;TURN MMU ON
9813 045642 106531              MFPI     @R1)+         ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
9814 045644 012604              MOV      (SP)+,R4      ;POP THE STACK
9815 045646 005037 177572      CLR      @0177572      ;TURN MMU OFF
9816 045652 020504              CMP      R5,R4         ;IS DATA EQUAL TO EXPECTED
9817 045654 001401              BEQ      4$            ;YES GO ON
9818 045656 104002              ERROR    +2            ;MMU ERROR
9819
9820 045660              4$:     JMP      T13FIN        ;NO IT IS AN ADDER ERROR
9821 045660 000167 000202
9822
9823      ; ADDER TEST PART A TABLES
9824
9825 045664 172240      ARAD1:  .WORD 172240
9826 045666 177642      .WORD 177642
9827 045670 172252      .WORD 172252
9828 045672 177640      .WORD 177640
9829 045674 172242      .WORD 172242
9830 045676 172254      .WORD 172254
9831 045700 177652      .WORD 177652
9832 045702 177644      .WORD 177644
9833 045704 172246      .WORD 172246
9834 045706 177654      .WORD 177654
9835 045710 172250      .WORD 172250
9836 045712 177660      .WORD 177660
9837 045714 000333      .WORD 333
9838 045716 000000      PARVA1: .WORD 000000
9839 045720 000010      .WORD 000010
9840 045722 177777      .WORD 177777
9841 045724 177601      .WORD 177601
9842 045726 000010      .WORD 000010
9843 045730 000052      .WORD 000052
9844 045732 000070      .WORD 000070
    
```

MEMORY MANAGEMENT TESTS

9845	045734	000010			.WORD	000010
9846	045736	000010			.WORD	000010
9847	045740	000060			.WORD	000060
9848	045742	000000			.WORD	000000
9849	045744	000010			.WORD	000010
9850	045746	000333			.WORD	333
9851	045750	000000	VIR1:		.WORD	000000
9852	045752	025000			.WORD	025000
9853	045754	135224			.WORD	135224
9854	045756	017700			.WORD	017700
9855	045760	033000			.WORD	033000
9856	045762	145252			.WORD	145252
9857	045764	121000			.WORD	121000
9858	045766	043000			.WORD	043000
9859	045770	075000			.WORD	075000
9860	045772	142000			.WORD	142000
9861	045774	117700			.WORD	117700
9862	045776	000111			.WORD	111
9863	046000	007000			.WORD	007000
9864	046002	000000	PHY1:		.WORD	000000
9865	046004	006000			.WORD	006000
9866	046006	015124			.WORD	015124
9867	046010	000000			.WORD	000000
9868	046012	014000			.WORD	014000
9869	046014	012452			.WORD	012452
9870	046016	010000			.WORD	010000
9871	046020	004000			.WORD	004000
9872	046022	016000			.WORD	016000
9873	046024	010000			.WORD	010000
9874	046026	017700			.WORD	017700
9875	046030	000111			.WORD	111
9876	046032	010000			.WORD	010000
9877	046034	010000	MODE1:		.WORD	010000
9878	046036	030000			.WORD	030000
9879	046040	010000			.WORD	010000
9880	046042	030000			.WORD	030000
9881	046044	010000			.WORD	010000
9882	046046	010000			.WORD	010000
9883	046050	030000			.WORD	030000
9884	046052	030000			.WORD	030000
9885	046054	010000			.WORD	010000
9886	046056	030000			.WORD	030000
9887	046060	010000			.WORD	010000
9888	046062	000111			.WORD	111
9889	046064	030000			.WORD	030000

9890				
9891	046066			
9902	046066			
9903				
9904				
9905	046066	032777	000400	133044
9906	046074	001405		
9907	046076	062737	000001	001204
9908				

```

;
T13FIN:
TS1822:
; TEST 22/18 BIT ADDRESS OPTION
;*****
;CHECK THE SOFTWARE SWITCH REGISTER TO DETERMINE IF THIS IS A 22 BIT OR AN
;18 BIT ADDRESS SYSTEM. BIT 08 IN THE SWR=1 INDICATES AN 18 BIT SYSTEM.
;IF WE'RE IN A 22 BIT SYSTEM WE CAN PERFORM SOME EXTRA TESTS.
BIT      #BIT08,#SWR      ;IS BIT 08 SET?
BEQ      100#             ;BRANCH IF ITS NOT
ADD      #1,#TESTN       ;KEEP TEST NUMBERS IN ORDER
;ADD 1 FOR THE TESTS WE'RE SKIPPING
    
```


MEMORY MANAGEMENT TESTS

```

9966 046332 012737 046374 000004 5$:  MOV    #6$,#04          ;SET UP FOR NXM TRAP
9967 046340 005067 131434          CLR    0              ;CLEAR ADDR 0
9968 046344 012767 020000 124002  MOV    #20000,KIPAR6  ;TEST ADDRESS BIT 19
9969 046352 012737 177777 140000  MOV    #177777,#140000 ;WRITE ALL ONES TO ADDR 2000000
9970 046360 005767 131414          TST    0              ;TEST ADDR 0. SHOULD= ZERO
9971 046364 001403          BEQ    6$            ;BRANCH IF ADDRESS 0=0
9972 046366 005067 131200          CLR    SRO           ;DISABLE MMU BEFORE ERROR
9973 046372 104002          CLR    SRO           ;MMU ERROR
9974          ERROR    +2      ;ERROR! BIT 19 DID NOT ASSERT
9975 046374 012737 046436 000004 6$:  MOV    #7$,#04          ;SET UP FOR NXM TRAP
9976 046402 005067 131372          CLR    0              ;CLEAR ADDR 0
9977 046406 012767 040000 123740  MOV    #40000,KIPAR6  ;TEST ADDRESS BIT 20
9978 046414 012737 177777 140000  MOV    #177777,#140000 ;WRITE ALL ONES TO ADDR 4000000
9979 046422 005767 131352          TST    0              ;TEST ADDR 0. SHOULD =0
9980 046426 001403          BEQ    7$            ;BRANCH IF ADDRESS 0 =0
9981 046430 005067 131136          CLR    SRO           ;DISABLE MMU BEFORE ERROR
9982 046434 104002          CLR    SRO           ;MMU ERROR
9983          ERROR    +2      ;ERROR! BIT 20 DID NOT ASSERT
9984 046436 012737 046500 000004 7$:  MOV    #8$,#04          ;SET UP FOR NXM
9985 046444 005067 131330          CLR    0              ;CLEAR ADDRESS 0
9986 046450 012767 100000 123676  MOV    #100000,KIPAR6 ;TEST ADDRESS BIT 21
9987 046456 012737 177777 140000  MOV    #177777,#140000 ;WRITE ALL ONES AT ADDR 10000000
9988 046464 005767 131310          TST    0              ;CHECK ADDRESS 0. SHOULD = 0
9989 046470 001403          BEQ    8$            ;BRANCH IF ADDR 0 = 0
9990 046472 005067 131074          CLR    SRO           ;DISABLE MMU BEFORE ERROR
9991 046476 104002          CLR    SRO           ;MMU ERROR
9992          ERROR    +2      ;ERROR! ADDR BIT 21 DID NOT ASSERT
9993 046500 005067 131066          CLR    SRO           ;DISABLE MMU
9994 046504 005037 177766          CLR    #0177766      ;CLEAR CPU ERROR REGISTER
9995 046510 012706 001000          MOV    #STBOT,R6     ;RESET STACK POINTER
9996 046514 013737 003012 000004  MOV    #0SLOC00,#04  ;RESTORE VECTOPS
9997 046522 013737 003014 000006  MOV    #0SLOC01,#06  ;
9998          ;
10007 046530          TSM14:
10008          ; ADDER RELOCATION TEST PART B
10009          ;*****
          ;(NEED 22 BITS OF MEMORY ADDRESSING)
10010 046530 005037 177572          CLR    #0SRO         ;TURN OFF MMU.
10011 046534 005067 131226          CLR    CPEREG        ;CLEAR THE CPU ERROR REGISTER
10012 046540 013737 000004 003012  MOV    #04,#0SLOC00  ;SAVE LOC 4 IN SLOC00.
10013 046546 013737 000006 003014  MOV    #06,#0SLOC01  ;SAVE LOC 6 IN SLOC01.
10014 046554 012737 047244 000004  MOV    #NXMTRP,#04   ;SET UP FOR TIMEOUT TRAP
10015 046562 012737 000340 000006  MOV    #340,#06      ;SET UP FOR TIMEOUT TRAP
10016 046570 005037 172340          CLR    #0KIPAR0      ;SET KER PAR0 FOR 1ST 4KW OF MEMORY.
10017 046574 012767 077406 123476  MOV    #77406,KIPDR0 ;SET KER PDR FOR 4KW R/W ACCESS.
10018 046602 012737 177500 172354  MOV    #177500,#0KIPAR6 ;SET UP KERNEL PAGE ADDR REG 6
10019          ;FOR HIGHEST 4K WORDS OF NON I/O
10020          ;FOR 2 MEG WORDS OF MEMORY.
10021 046610 012767 077406 123476  MOV    #77406,KIPDR6 ;SET KER PDR6 FOR 4KW R/W ACCESS.
10022 046616 012737 000020 172516  MOV    #20,#0SR3     ;ENABLE 22 BIT ADDRESSING.
10023 046624 012737 000001 177572  MOV    #1,#0SRO      ;TURN ON THE MMU.
10024 046632 005737 147776          TST    #0147776     ;ATTEMPT TO ADDR. LAST MEMORY ADDR. $$$
10025          ;*****WILL TRAP TO 4 IF 2 MEG WORDS OF MEMORY NOT AVAILABLE*****
10026 046636 013737 003012 000004  MOV    #0SLOC00,#04  ;RESTORE LOC 4
10027 046644 013737 003014 000006  MOV    #0SLOC01,#06  ;RESTORE LOC 6
10028 046652 005037 177572          CLR    #0177572     ;MMU OFF
10029 046656 005037 003030          CLR    #0FLAG       ;CLEAR MMU ABORT FLAG

```


MEMORY MANAGEMENT TESTS

```

10087 ;ROUTINE TO WRITE DATA TO PHYSICAL ADDRESS AND TO CHECK IF DATA AT
10088 ;PHYSICAL ADDRESS IS EQUAL TO EXPECTED AND IF NOT DETERMINE IF IT IS
10089 ;AN ADDER ERROR OR A MEMORY ERROR
10090 ;
10091 047134 012437 177776 TS14: MOV (R4)+,0#177776 ;INIT PSW
10092 047140 012737 000001 177572 MOV #1,0#177572 ;TURN MMU ON
10093 047146 012746 052525 MOV #52525,-(SP) ;WRITE DATA ONTO STACK
10094 047152 006631 MTP1 0(R1)+ ;WRITE DATA TO PHYSICAL ADDRESS IN A STACK
10095 047154 005037 177572 CLR 0#177572 ;TURN MMU OFF
10096 047160 012737 010000 177776 T14: MOV #10000,0#177776 ;INIT PSW
10097 047166 012237 172246 MOV (R2)+,0#172246 ;INIT SIPAR3
10098 047172 012737 000001 177572 MOV #1,0#177572 ;TURN MMU ON
10099 047200 006573 000000 MFPI 00(R3) ;DO RELOCATION
10100 047204 022726 052525 CMP #52525,(SP)+ ;IS DATA EQUAL TO EXPECTED
10101 047210 001410 BEQ 2# ;YES GO ON
10102 047212 006573 000000 MFPI 00(R3) ;WHAT TYPE OF ERROR IS IT
10103 047216 022726 125252 CMP #125252,(SP)+ ;
10104 047222 001402 BEQ 1# ;
10105 047224 104002 ERROR +2 ;MMU ERROR
10106 ;IT IS A MEMORY ERROR
10107 047226 000401 BR 2# ;
10108 047230 104002 1#: ERROR +2 ;MMU ERROR
10109 ;IT IS AN ADDER ERROR
10110 047232 005037 177572 2#: CLR 0#177572 ;TURN MMU OFF
10111 047236 005203 INC R3 ;
10112 047240 005203 INC R3 ;
10113 047242 000207 RTS PC ;RETURN
10114 ;
10115 ;NON-EXISTANT MEMORY TRAP ROUTINE
10116 ;
10117 047244 005037 177572 NXMTRP: CLR 0#SRO ;TURN OFF MMU.
10118 047250 012716 047532 MOV #T14FIN,(SP) ;SET UP STACK WITH RETURN ADDR.
10119 047254 013737 003012 000004 MOV 0#SLOC00,0#4 ;RESTORE LOC 4
10120 047262 013737 003014 000006 MOV 0#SLOC01,0#6 ;RESTORE LOC 6
10121 047270 005037 177766 CLR 0#177766 ;CLEAR TIME OUT INDICATION FROM
10122 ;CPU ERROR REGISTER.
10123 047274 000006 RTT ;RETURN FROM TRAP; GO TO NEXT TEST.
10124 ;
10125 ;ADDER TEST PART B TABLES
10126 ;
10127 047276 177646 PARAD2: .WORD 177646
10128 047300 177650 .WORD 177650
10129 047302 177652 .WORD 177652
10130 047304 172240 .WORD 172240
10131 047306 177640 .WORD 177640
10132 047310 177642 .WORD 177642
10133 047312 172244 .WORD 172244
10134 047314 177644 .WORD 177644
10135 047316 172252 .WORD 172252
10136 047320 172352 .WORD 172352
10137 047322 177662 .WORD 177662
10138 047324 172242 .WORD 172242
10139 047326 000333 .WORD 333
10140 047330 157700 PARVA2: .WORD 157700
10141 047332 137700 .WORD 137700
10142 047334 077700 .WORD 077700
10143 047336 176777 .WORD 176777
    
```

MEMORY MANAGEMENT TESTS

10144	047340	007600	.WORD	007600
10145	047342	167700	.WORD	167700
10146	047344	175700	.WORD	175700
10147	047346	177425	.WORD	177425
10148	047350	177220	.WORD	177220
10149	047352	173700	.WORD	173700
10150	047354	176700	.WORD	176700
10151	047356	077400	.WORD	077400
10152	047360	000333	.WORD	333
10153	047362	070000	VIR2: .WORD	070000
10154	047364	110000	.WORD	110000
10155	047366	130000	.WORD	130000
10156	047370	000000	.WORD	000000
10157	047372	000000	.WORD	000000
10158	047374	030000	.WORD	030000
10159	047376	050000	.WORD	050000
10160	047400	052524	.WORD	052524
10161	047402	136000	.WORD	136000
10162	047404	130000	.WORD	130000
10163	047406	000111	.WORD	111
10164	047410	030000	.WORD	030000
10165	047412	030000	.WORD	030000
10166	047414	030000	MODE2: .WORD	030000
10167	047416	030000	.WORD	030000
10168	047420	030000	.WORD	030000
10169	047422	010000	.WORD	010000
10170	047424	030000	.WORD	030000
10171	047426	030000	.WORD	030000
10172	047430	010000	.WORD	010000
10173	047432	030000	.WORD	030000
10174	047434	010000	.WORD	010000
10175	047436	000000	.WORD	000000
10176	047440	000111	.WORD	111
10177	047442	030000	.WORD	030000
10178	047444	010000	.WORD	010000
10179	047446	160000	PARVA3: .WORD	160000
10180	047450	140000	.WORD	140000
10181	047452	100000	.WORD	100000
10182	047454	176770	.WORD	176770
10183	047456	007600	.WORD	007600
10184	047460	170000	.WORD	170000
10185	047462	176000	.WORD	176000
10186	047464	177552	.WORD	177552
10187	047466	177400	.WORD	177400
10188	047470	174000	.WORD	174000
10189	047472	177000	.WORD	177000
10190	047474	007500	.WORD	007500
10191	047476	000333	.WORD	333
10192	047500	060000	VIR3: .WORD	060000
10193	047502	060000	.WORD	060000
10194	047504	060000	.WORD	060000
10195	047506	060700	.WORD	060700
10196	047510	060000	.WORD	060000
10197	047512	060000	.WORD	060000
10198	047514	060000	.WORD	060000
10199	047516	060024	.WORD	060024
10200	047520	060000	.WORD	060000

MEMORY MANAGEMENT TESTS

```

10201 047522 060000 .WORD 060000
10202 047524 060000 .WORD 060000
10203 047526 060000 .WORD 060000
10204 047530 000333 .WORD 333
10205
10206
10207
10208 047532 ;
;T14FIN:
;
; TEST NON-EXISTANT MEMORY TRAP
;*****
;WE ARE ASSUMING THAT THE NON-EXISTANT MEMORY TIME OUT
;FEATURE IS WORKING SINCE WE CAN'T GUARANTEE THAT
;THE SYSTEM BEING TESTED HAS A NO-I-EXISTANT MEMORY LOCATION.
;AT THIS TIME WE WILL ATTEMPT TO TEST THE NXM FUNCTION
10224 047532 004767 070014 JSR PC,MMU ;INIT THE MMU
10225 047536 012737 177400 172354 MOV #177400,#KIPAR6 ;SET KIPAR6 TO RELOCATE TO HIGHEST MEMORY
10226 047544 016767 130231 133240 MOV 4,SLOC00 ;SAVE VECTOR
10227 047552 016767 000026 130224 MOV 2#,4 ;LOAD VEC WITH ADDR OF TRAP HANDLER
10228 047560 052767 000001 130004 BIS #BIT00,SRO ;TURN ON THE MMU
10229 047566 005067 130174 CLR CPREG ;CLEAR THE CPU ERROR REGISTER
10230 047572 005067 130200 CLR PS ;CLEAR THE PSW
10231 047576 005737 157776 TST #157776 ;ACCESS PHYSICAL ADDR 17757776
10232 047602 000415 1#: BR NXMFIN ;IF IT DOESN'T TRAP WE'LL ASSUME
10233 ;THAT THIS IA A 4 MEGABYTE SYSTEM
10234 ;AND GO TO THE NEXT TEST
10235 047604 022767 000040 130154 2#: CMP #BIT05,CPREG ;IS CPU ERROR REGISTER CORRECT?
10236 047612 001401 BEQ 3# ;
10237 047614 104002 ERROR +2 ;MMU ERROR
10238 047616 022726 047602 3#: CMP #1#,(SP)+ ;IS CONTENTS OF STACK CORRECT?
10239 047622 001401 BEQ 4# ;
10240 047624 104002 ERROR +2 ;MMU ERROR
10241 047626 022726 000000 4#: CMP #0,(SP)+ ;IS CONTENTS OF STACK CORRECT?
10242 047632 001401 BEQ NXMFIN ;
10243 047634 104002 ERROR +2 ;MMU ERROR
10244 047636 005067 127730 NXMFIN: CLR SRO ;TURN OFF THE MMU
10245 047642 005067 130120 CLR CPREG ;CLEAR THE CPU ERROR REGISTER
10246 047646 016767 133140 130130 MOV SLOC00,4 ;RESTORE THE VECTOR
10247
10249
10251 047614 TSMM15:
10252 ; PAGE WRITTEN BIT TEST
10253 047654 005037 177572 CLR #177572 ;MMU OFF
10254 047660 005067 133144 CLR FLAG ;CLEAR MMU ABORT FLAG
10255 047664 004767 067662 JSR PC,MMU ;INIT MMU
10256 047670 005037 177776 CLR #177776 ;INIT PSW
10257 047674 012704 172300 MOV #172300,R4 ;SET POINTER TO KPDRS
10258 047700 004767 000114 JSR PC,TS15 ;DO RELOCATIONS AND TEST KIPDRS FOR
10259 ;PAGE WRITTEN BIT BEING SET AND IF
10260 ;NOT SET GO TO ERROR
10261 047704 004767 000160 JSR PC,T15 ;DO RELOCATIONS AND TEST KPDRS FOR
10262 ;PAGE WRITTEN BIT BEING SET AND IF NOT
10263 ;SET GO TO ERROR
10264 047710 012737 050000 177776 MOV #50000,#177776 ;INIT PSW
10265 047716 012704 172200 MOV #172200,R4 ;SET POINTER TO SPDRS
10266 047722 004767 000072 JSR PC,TS15 ;DO RELOCATIONS AND TEST SIPDRS FOR
10267 ;PAGE WRITTEN BIT BEING SET AND IF NOT

```

MEMORY MANAGEMENT TESTS

```

10268
10269 047726 004767 000136          JSR    PC,T15          ;SET GO TO ERROR
10270                                     ;DO RELOCATIONS AND TEST SDPRS FOR
10271                                     ;PAGE WRITTEN BIT BEING SET AND IF NOT
10272 047732 005037 177776          CLR    @#177776       ;SET GO TO ERROR
10273 047736 012737 170000 177776   MOV    @#170000,@#177776 ;INIT PSW TO A KNOWN STATE
10274 047744 012704 177600          MOV    @#177600,R4    ;INIT PSW
10275 047750 004767 000044          JSR    PC,TS15        ;SET POINTER TO UPDRS
10276                                     ;DO RELOCATIONS AND TEST UIPDRS FOR
10277                                     ;PAGE WRITTEN BIT BEING SET AND IF
10278 047754 004767 000110          JSR    PC,T15        ;NOT SET GO TO ERROR
10279                                     ;DO RELOCATIONS AND TEST UDPDRS FOR
10280                                     ;PAGE WRITTEN BIT BEING SET AND IF NOT
10281 047760 005037 177776          CLR    @#177776       ;SET GO TO ERROR
10282 047764 012704 172300          MOV    @#172300,R4    ;INIT PSW TO A KNOWN STATE
10283 047770 004767 000152          JSR    PC,T15A       ;SET POINTER TO KPDRS
10284                                     ;EXPLICITLY WRITE TO KPDRS AND TEST
10285                                     ;FOR PAGE WRITTEN BIT BEING CLEARED
10286 047774 012704 172200          MOV    @#172200,R4    ;AND IF NOT CLEARED GO TO ERROR
10287 050000 004767 000142          JSR    PC,T15A       ;SET POINTER TO SPDRS
10288                                     ;EXPLICITLY WRITE TO SPDRS AND TEST
10289                                     ;FOR PAGE WRITTEN BIT BEING CLEARED
10290 050004 012704 177600          MOV    @#177600,R4    ;AND IF NOT CLEARED GO TO ERROR
10291 050010 004767 000132          JSR    PC,T15A       ;SET POINTER TO UPDRS
10292                                     ;EXPLICITLY WRITE TO UPDRS AND TEST
10293                                     ;FOR PAGE WRITTEN BIT BEING CLEARED
10294                                     ;AND IF NOT CLEARED GO TO ERROR
10295 050014 000167 000154          JMP    T15FIN
10296
10297          ;ROUTINE TO DO RELOCATIONS AND TEST IPDRS FOR PAGE WRITTEN BIT BEING
10298          ;SET AND IF NOT SET REPORT AN ERROR
10299
10300 050020 005001          TS1:  CLR    R1          ;SET POINTER TO VIRTUAL ADDRESS
10301 050022 012737 000020 172516   MOV    @#20,@#172516  ;INIT MMR3
10302 050030 012737 000001 177572 1$:  MOV    @#1,@#177572  ;TURN MMU ON
10303 050036 011111          MOV    (R1),(R1)     ;DO A RELOCATION
10304 050040 005037 177572          CLR    @#177572     ;TURN MMU OFF
10305 050044 022427 077506          CMP    (R4)+,@#77506 ;IS DATA EQUAL TO EXPECTED
10306 050050 001401          BEQ    2$           ;OK GO ON
10307 050052 104002          ERROR +2           ;MMU ERROR
10308                                     ;NO GO TO ERROR
10309 050054 062701 020000 2$:  ADD    @#20000,R1    ;POINT TO NEXT VIRTUAL ADDRESS
10310 050060 020127 160000          CMP    R1,@#160000  ;ARE WE DONE
10311 050064 001361          BNE    1$           ;NO GO TO 1$
10312 050066 000207          RTS    PC           ;RETURN
10313
10314          ;ROUTINE TO DO RELOCATIONS AND TEST DPDRS FOR PAGE WRITTEN BIT BEING SET
10315          ;AND IF NOT SET REPORT AN ERROR
10316
10317 050070 005001          T15:  CLR    R1          ;SET POINTER TO VIRTUAL ADDRESS
10318 050072 062704 000002          ADD    @#2,R4        ;POINT TO FIRST DPDR
10319 050076 012737 000027 172516   MOV    @#27,@#172516 ;INIT MMR3
10320 050104 012737 000001 177572 1$:  MOV    @#1,@#177572  ;TURN MMU ON
10321 050112 011146          MOV    (R1),-(SP)   ;PUSH DATA ONTO THE STACK
10322 050114 106611          MTPD  (R1)         ;DO A RELOCATION
10323 050116 005037 177572          CLR    @#177572     ;TURN MMU OFF
10324 050122 022427 077506          CMP    (R4)+,@#77506 ;IS DATA EQUAL TO EXPECTED

```

MEMORY MANAGEMENT TESTS

```

10325 050126 001401      BEQ      2$      ;OK GO ON
10326 050130 104002      ERROR    +2      ;MMU ERROR
10327                      ;NO GO TO ERROR
10328 050132 062701 020000 2$:      ADD      @20000,R1 ;POINT TO NEXT VIRTUAL ADDRESS
10329 050136 020127 160000      CMP      R1,@160000 ;ARE WE DONE
10330 050142 001360      BNE      1$      ;NO GO TO 1$
10331 050144 000207      RTS      PC      ;RETURN
10332                      ;
10333                      ;ROUTINE TO EXPLICITLY WRITE TO PDRS AND TEST PAGE WRITTEN BIT FOR BEING
10334                      ;CLEARED AND IF NOT CLEARED REPORT AN ERROR
10335                      ;
10336 050146 005002      T15A:  CLR      R2      ;CLEAR COUNTER
10337 050150 011414      1$:      MOV      (R4),(R4) ;DO AN EXPLICIT WRITE TO PDR
10338 050152 022427 077406      CMP      (R4)+,@77406 ;IS DATA EQUAL TO EXPECTED
10339 050156 001401      BEQ      2$      ;OK GO ON
10340 050160 104002      ERROR    +2      ;MMU ERROR
10341                      ;NO GO TO ERROR
10342 050162 005202      2$:      INC      R2      ;INCREMENT POINTER
10343 050164 020227 000020      CMP      R2,@20    ;ARE WE DONE
10344 050170 001367      BNE      1$      ;NO GO TO 1$
10345 050172 000207      RTS      PC      ;RETURN
10346 050174                      T15FIN:
10347                      ;
10350 050174                      TSM16:
10351                      ;
10352 050174 005037 177572      ; TEST CSM (CALL SUPERVISOR MODE)
10353 050200 005037 003030      CLR      @177572 ;MMU OFF
10354 050204 012704 050524      CLR      @FLAG   ;CLEAR MMU ABORT FLAG
10355 050210 004767 067336      MOV      @TMM16E,R4 ;INIT R4
10356 050214 012737 000037 172516 JSP      PC,MMU   ;INIT MMU
10357 050222 005037 177776      MOV      @37,@172516 ;ENABLE CSM INSTRUCTION
10358 050226 013746 000010      CLR      @177776 ;SET PS TO KER MODE
10359 050232 013746 000014      MOV      @10,-(SP) ;SAVE VECTORS
10360 050236 013746 000016      MOV      @14,-(SP) ;
10361 050242 012737 050414 000010      MOV      @16,-(SP) ;
10362 050250 012737 000137 000014      MOV      @TMM16B,@10 ;SETUP NEW VECTORS
10363 050256 012737 050534 000016      MOV      @137,@14 ;
10364 050264 007014      MOV      @TMM16A,@16 ;
10365 050266 104002      .WORD   7014      ; TEST INSTRUCTION
10366                      ERROR    +2      ;MMU ERROR
10367 050270 012737 050444 000010 TSM16A: MOV      @TMM16C,@10 ;GO TO ERROR IF NOT TRAPPED
10368 050276 012737 000027 172516      MOV      @27,@172516 ;SETUP NEW VECTOR
10369 050304 012737 140000 177776      MOV      @140000,@177776 ;DISABLE CSM INSTRUCTION
10370 050312 007014      .WORD   7014      ;SET PS TO USER MODE
10371 050314 104002      ERROR    +2      ; TEST INSTRUCTION
10372                      ;MMU ERROR
10373 050316 012737 050474 000010 TSM16B: MOV      @TMM16D,@10 ;GO TO ERROR IF NOT TRAPPED
10374 050324 012737 000027 172516      MOV      @27,@172516 ;SETUP NEW VECTOR
10375 050332 005037 177776      CLR      @177776 ;DISABLE CSM INSTRUCTION
10376 050336 007014      .WORD   7014      ;SET PS TO KER MODE
10377 050340 104002      ERROR    +2      ; TEST INSTRUCTION
10378                      ;MMU ERROR
10379 050342 012737 000037 172516 TSM16C: MOV      @37,@172516 ;GO TO ERROR IF NOT TRAPPED
10380 050350 012737 040000 177776      MOV      @40000,@177776 ;ENABLE CSM INSTRUCTION
10381 050356 012706 000700      MOV      @700,R6 ;SET PS TO SUP MODE
10382 050362 012737 140000 177776      MOV      @140000,@177776 ;INIT SUP SP
10383 050370 012706 000600      MOV      @600,R6 ;SET PS TO USER MODE
                      ;INIT USER SP

```

MEMORY MANAGEMENT TESTS

```

10384 050374 012737 000014 000010      MOV      #14,0#10      ;SETUP NEW VECTOR
10385 050402 000277                      SCC                      ;SET ALL CC BITS
10386 050404 007024                      .WORD    7024          ; TEST INSTRUCTION
10387 050406 104002                      TSM16D: ERROR +2      ;MMU ERROR
10388                                     ;GO TO ERROR IF NOT TRAPPED
10389 050410 000167 000504                      JMP      TM16A
10390                                     ;
10391                                     ;
10392 050414 042737 007777 177776 TMM16B: BIC      #7777,0#177776 ;CLEAR UNWANTED BITS
10393 050422 022737 000000 177776      CMP      #0,0#177776 ;IS PS CORRECT
10394 050430 001401                      BEQ     1#             ;YES GO ON
10395 050432 104002                      ERROR   +2            ;MMU ERROR
10396                                     ;NO GO TO ERROR
10397 050434 005726                      1#:    TST      (SP)+ ;CLEAN UP STACK
10398 050436 005726                      TST      (SP)+ ;
10399 050440 000167 177624                      JMP      TSM16A ;
10400 050444 042737 007777 177776 TMM16C: BIC      #7777,0#177776 ;CONTINUE TESTING
10401 050452 022737 030000 177776      CMP      #30000,0#177776 ;CLEAR UNWANTED BITS
10402 050460 001401                      BEQ     1#             ;IS PS CORRECT
10403 050462 104002                      ERROR   +2            ;YES GO ON
10404                                     ;MMU ERROR
10405 050464 005726                      1#:    TST      (SP)+ ;NO GO TO ERROR
10406 050466 005726                      TST      (SP)+ ;CLEAN UP STACK
10407 050470 000167 177622                      JMP      TSM16B ;
10408 050474 042737 007777 177776 TMM16D: BIC      #7777,0#177776 ;CONTINUE TESTING
10409 050502 022737 000000 177776      CMP      #0,0#177776 ;CLEAR UNWANTED BITS
10410 050510 001401                      BEQ     1#             ;IS PS CORRECT
10411 050512 104002                      ERROR   +2            ;YES GO ON
10412                                     ;MMU ERROR
10413 050514 005726                      1#:    TST      (SP)+ ;NO GO TO ERROR
10414 050516 005726                      TST      (SP)+ ;CLEAN UP STACK
10415 050520 000167 177616                      JMP      TSM16C ;
10416 050524 156430                      TMM16E: .WORD    156430 ;CONTINUE TESTING
10417 050526 104002                      TMM16F: ERROR   +2    ; TEST LOCATION
10418                                     ;MMU ERROR
10419 050530 000167 000364                      JMP      TM16A ;GO TO ERROR IF DIDN'T ABORT
10420 050534 022737 070017 177776 TMM16A: CMP      #70017,0#177776 ;IS PS CORRECT
10421 050542 001401                      BEQ     1#             ;YES GO ON
10422 050544 104002                      ERROR   +2            ;MMU ERROR
10423                                     ;NO GO TO ERROR
10424 050546 020627 000572                      1#:    CMP      R6,#572 ;IS SP CORRECT
10425 050552 001401                      BEQ     2#             ;YES GO ON
10426 050554 104002                      ERROR   +2            ;MMU ERROR
10427                                     ;NO GO TO ERROR
10428 050556 020427 050526                      2#:    CMP      R4,#TMM16E+2 ;IS R4 CORRECT
10429 050562 001401                      BEQ     3#             ;YES GO ON
10430 050564 104002                      ERROR   +2            ;MMU ERROR
10431                                     ;NO GO TO ERROR
10432 050566 023727 050524 156430 3#:    CMP      #TMM16E,#156430 ;IS TEST LOCATION OK
10433 050574 001401                      BEQ     4#             ;YES GO ON
10434 050576 104002                      ERROR   +2            ;MMU ERROR
10435                                     ;NO GO TO ERROR
10436 050600 022627 156430                      4#:    CMP      (SP)+,#156430 ;IS STACK CORRECT
10437 050604 001401                      BEQ     5#             ;YES GO ON
10438 050606 104002                      ERROR   +2            ;MMU ERROR
10439                                     ;NO GO TO ERROR
10440 050610 022627 050406                      5#:    CMP      (SP)+,#TSM16D ;IS STACK CORRECT

```

MEMORY MANAGEMENT TESTS

10441	050614	001401				BEQ	6:				;YES GO ON
10442	050616	104002				ERROR	+2			;MMU ERROR	
10443										;NO GO TO ERROR	
10444	050620	022627	140000		6:	CMP	(SP)+, #140000			;IS STACK CORRECT	
10445	050624	001401				BEQ	7:			;YES GO ON	
10446	050626	104002				ERROR	+2			;MMU ERROR	
10447										;NO GO TO ERROR	
10448	050630	012706	000700		7:	MOV	#700, R6			;RESTORE SUP SP	
10449	050634	012737	140000	177776		MOV	#140000, #177776			;SET PS TO USER MODE	
10450	050642	020627	000600			CMP	R6, #600			;IS USER SP CORRECT	
10451	050646	001401				BEQ	8:			;YES GO ON	
10452	050650	104002				ERROR	+2			;MMU ERROR	
10453										;NO GO TO ERROR	
10454	050652	012767	077400	121320	8:	MOV	#77400, SIPDRO			;SETUP SIPDRO TO ABORT	
10455	050660	012737	050526	000016		MOV	#TMM16F, #16			;SETUP VECTOR	
10456	050666	012737	000001	003030		MOV	#1, #FLAG			;SETUP FLAG FOR AN ABORT	
10457	050674	012737	000001	177572		MOV	#1, #177572			;TURN MMU ON	
10458	050702	010701				MOV	R7, R1			;SAVE OLD PC	
10459	050704	007014				.WORD	7014			;TEST INSTRUCTION	
10460	050706	022737	000000	003030		CMP	#0, #FLAG			;DID AN ABORT OCCUR	
10461	050714	001401				BEQ	9:			;YES GO ON	
10462	050716	104002				ERROR	+2			;MMU ERROR	
10463										;NO GO TO ERROR	
10464	050720	023701	003046		9:	CMP	#SAVMR2, R1			;IS MMR2 CORRECT	
10465	050724	001401				BEQ	10:			;YES GO ON	
10466	050726	104002				ERROR	+2			;MMU ERROR	
10467										;NO GO TO ERROR	
10468	050730	023727	003042	100041	10:	CMP	#SAVMR0, #100041			;IS MMRO CORRECT	
10469	050736	001401				BEQ	11:			;YES GO ON	
10470	050740	104002				ERROR	+2			;MMU ERROR	
10471										;NO GO TO ERROR	
10472	050742	012737	000037	172516	11:	MOV	#37, #172516			;ENABLE CSM	
10473	050750	012737	040000	177776		MOV	#40000, #177776			;SET PSW TO SUP	
10474	050756	012706	000700			MOV	#700, R6			;SETUP SUP SP	
10475	050762	012737	140000	177776		MOV	#140000, #177776			;SET PSW TO USE	
10476	050770	012706	000600			MOV	#600, R6			;SETUP USE SP	
10477	050774	012737	000014	000010		MOV	#14, #10			;SETUP NEW VECTOR	
10478	051002	012737	051024	000016		MOV	#TS16, #16			;SETUP NEW VECTOR	
10479	051010	000277				SCC				;SET ALL CC BITS	
10480	051012	007027				.WORD	7027			;TEST INSTRUCTION	
10481	051014	045712				.WORD	45712				
10482	051016	104002			TS16A:	ERROR	+2			;MMU ERROR	
10483										;GO TO ERROR IF DIDN'T TRAP	
10484	051020	000167	000074			JMP	TM16A				
10485	051024	022737	070017	177776	TS16:	CMP	#70017, #177776			;IS PSW CORRECT	
10486	051032	001401				BEQ	200:			;YES GO ON	
10487	051034	104002				ERROR	+2			;MMU ERROR	
10488										;NO GO TO ERROR	
10489	051036	020627	000572		200:	CMP	R6, #572			;IS SP CORRECT	
10490	051042	001401				BEQ	201:			;YES GO ON	
10491	051044	104002				ERROR	+2			;MMU ERROR	
10492										;NO GO TO ERROR	
10493	051046	022627	045712		201:	CMP	(SP)+, #45712			;IS STACK CORRECT	
10494	051052	001401				BEQ	202:			;YES GO ON	
10495	051054	104002				ERROR	+2			;MMU ERROR	
10496										;NO GO TO ERROR	
10497	051056	022627	051016		202:	CMP	(SP)+, #TS16A			;IS STACK CORRECT	

MEMORY MANAGEMENT TESTS

```

10498 051062 001401          BEQ      203$          ;YES GO ON
10499 051064 104002          ERROR    +2           ;MMU ERROR
10500                                ;NO GO TO ERROR
10501 051066 022627 140000    203$:  CMP      (SP)+, #140000 ;IS STACK CORRECT
10502 051072 001401          BEQ      204$          ;YES GO ON
10503 051074 104002          ERROR    +2           ;MMU ERROR
10504                                ;NO GO TO ERROR
10505 051076 012706 000700    204$:  MOV      #700,R6      ;RESTORE SUP SP
10506 051102 012737 140000    177776 MOV      #140000,#177776 ;SET PSW TO USER MODE
10507 051110 020627 000600    CMP      R6,#600      ;IS USER SP CORRECT
10508 051114 001401          BEQ*    TM16A         ;YES GO ON
10509 051116 104002          ERROR    +2           ;MMU ERROR
10510                                ;NO GO TO ERROR
10511 051120 005037 177776    TM16A: CLR      #177776   ;SET PS TO KER MODE
10512 051124 012637 000016    MOV      (SP)+,#16    ;RESTORE VECTORS
10513 051130 012637 000014    MOV      (SP)+,#14    ;
10514 051134 012637 000010    MOV      (SP)+,#10    ;
10515
10519
10520                                ;*****
10521                                ;.SBTTL FLOATING POINT TESTS
10522                                ;*****
10523                                ;          BEGIN FLOATING POINT TESTING
10524                                ;*****
10537 051140                    MBT1:
10538
10539                                ;          FPP REGISTER BIT TESTS
10540                                ;*****
                                ;R5=FPP POINTER
                                ;R1=TEMPORARY COUNTER
                                ;R2=POINTER TO EXPECTED DATA
                                ;R3=POINTER TO RECEIVED DATA
                                ;R4=ODD/EVEN COUNTER
10541 051140 170011          SETD
10542 051142 005005                    MBT2:  CLR      R5          ;SETUP FPP ACC POINTER
10543 051144 012702 003100    MOV      #BTEXP,R2    ;POINT TO TEST DATA
10544 051150 012703 003110    MOV      #BTRES,R3    ;POINT TO RECEIVED DATA
10545 051154 170400                    MBT2A: CLRD     AC0       ;SETUP FPP REGISTER VALUES
10546 051156 174012          STD      AC0,(R2)     ;CLEAR EXPECTED VALUE
10547 051160 005004          CLR      R4
10548 051162 170400                    BTGO:  CLRD     AC0       ;SETUP FPP REGISTER VALUES
10549 051164 170401          CLRD     AC1
10550 051166 170402          CLRD     AC2
10551 051170 170403          CLRD     AC3
10552 051172 170404          CLRD     AC4
10553 051174 170405          CLRD     AC5
10554
10555 051176 010501          MOV      R5,R1        ;GET FPP AC NUMBER INTO R1
10556 051200 070127 000014    MUL      #14,R1        ;ALLOW 10 LOCATIONS FOR OPERATION
10557 051204 062701 051212    ADD      #MAC0,R1      ;SETUP JMP LOCATION
10558 051210 000111          JMP      (R1)
10559 051212 172467 131662    MAC0:  LDD      BTEXP,AC0 ;LOAD TEST DATA INTO TEST REGISTER
10560 051216 174067 131666    MAC0A: STD      AC0,BTRES ;SAVE TEST RESULT
10561 051222 000167 000074    JMP
10562 051226 172567 131646    MAC1:  LDD      BTEXP,AC1 ;LOAD TEST DATA INTO TEST REGISTER
10563 051232 174167 131652    STD      AC1,BTRES    ;SAVE TEST RESULT
10564 051236 000167 000060    JMP      MACE         ;GET OUT

```


FLOATING POINT TESTS

```

10565 051242 172667 131632      MAC2:  LDD      BTEXP,AC2      ;LOAD TEST DATA INTO TEST REGISTER
10566 051246 174267 131636      STD      AC2,BTRES      ;SAVE TEST RESULT
10567 051252 000167 000044      JMP      MACE           ;GET OUT
10568 051256 172767 131616      MAC3:  LDD      BTEXP,AC3      ;LOAD TEST DATA INTO TEST REGISTER
10569 051262 174367 131622      STD      AC3,BTRES      ;SAVE TEST RESULT
10570 051266 000167 000030      JMP      MACE           ;GET OUT
10571 051272 172467 131602      MAC4:  LDD      BTEXP,AC0      ;LOAD TEST DATA INTO TEST REGISTER
10572 051276 174004      STD      AC0,AC4      ;SAVE TEST RESULT
10573 051300 172404      LDD      AC4,AC0      ;GET OUT
10574 051302 000167 177710      JMP      MAC0A         ;LOAD TEST DATA INTO TEST XFER REGISTER
10575 051306 172467 131566      MAC5:  LDD      BTEXP,AC0      ;LOAD TEST REGISTER
10576 051312 174005      STD      AC0,AC5      ;STORE RESULT INTO XFER FPP REGISTER
10577 051314 172405      LDD      AC5,AC0      ;GET OUT
10578 051316 000167 177674      JMP      MAC0A
10579 051322 026767 131552 131560 MACE:  CMP      BTEXP,BTRES      ;BRANCH IF REGISTER ERROR
10580 051330 001014      BNE     BTER
10581 051332 026767 131544 131552      CMP      BTEXP+2,BTRES+2
10582 051340 001010      BNE     BTER
10583 051342 026767 131536 131544      CMP      BTEXP+4,BTRES+4
10584 051350 001004      BNE     BTER
10585 051352 026767 131530 131536      CMP      BTEXP+6,BTRES+6
10586 051360 001403      BEQ     MBT8           ;GOOD RESULT
10587 051362 004737 140132      BTER:  CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
10588 051366 104003      ERROR   +3           ;FPP ERROR
10589
10590      ;NOW VERIFY THE OTHER REGISTERS REMAINED ZERO
10591 051370      MBT8:
10592 051370 005001      CLR     R1           ;CLEAR TEMPORARY COUNTER
10593 051372 005705      TST     R5           ;SEE IF R0 UNDER TEST
10594 051374 001413      BEQ     MBT8A        ;BRANCH IF TEST ING R0
10595 051376 020527 000004      CMP     R5,#4        ;SEE IF TESTING FPP REGISTER >R4
10596 051402 100010      BPL     MBT8A        ;SKIP R0 TESTING
10597 051404 174067 131500      STD     AC0,BTRES      ;SAVE AC TEST RESULT
10598 051410 004767 000246      JSR     R7,BTTST      ;VERIFY THAT CONTENTS REMAINED ZERO
10599 051414 001403      BEQ     MBT8A        ;BRANCH IF EXPECTED RESULT
10600 051416 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
10601 051422 104003      ERROR   +3           ;FPP ERROR
10602
10603 051424 020527 000001      MBT8A: CMP     R5,#1        ;SEE IF R1 UNDER TEST
10604 051430 001410      BEQ     MBT8B        ;BRANCH IF R1 UNDER TEST
10605 051432 174167 131452      STD     AC1,BTRES      ;SAVE AC TEST RESULT
10606 051436 004767 000220      JSR     R7,BTTST      ;VERIFY THAT CONTENTS REMAINED ZERO
10607 051442 001403      BEQ     MBT8B        ;BRANCH IF GOOD
10608 051444 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
10609 051450 104003      ERROR   +3           ;FPP ERROR
10610
10611 051452 020527 000002      MBT8B: CMP     R5,#2        ;SEE IF TESTING FPP REGISTER AC2
10612 051456 001410      BEQ     MBT8C        ;BRAN + IF R2 UNDER TEST
10613 051460 174267 131424      STD     AC2,BTRES      ;SAVE .C TEST RESULT
10614 051464 004767 000172      JSR     R7,BTTST      ;VERIFY THAT CONTENTS REMAINED ZERO
10615 051470 001403      BEQ     MBT8C        ;BRANCH IF GOOD
10616 051472 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $
10617 051476 104003      ERROR   +3           ;FPP ERROR
10618
10619 051500 020527 000003      MBT8C: CMP     R5,#3        ;SEE IF R3 UNDER TEST
10620 051504 001410      BEQ     MBT8D        ;BRANCH IF R3 UNDER TEST
10621 051506 174367 131376      STD     AC3,BTRES      ;SAVE AC TEST RESULT

```


FLOATING POINT TESTS

```

10690
10691
10692
; TEST UNIQUENESS OF FPP ACCUMULATORS
;*****
;THIS TEST LOADS UNIQUE PATTERNS INTO EACH ACCUMULATOR SIMULTANEOUSLY.
;R2=POINTER TO EXPECTED DATA
;R3=POINTER TO RECEIVED DATA

10693
10694 051712 170011
10695 051714 005000
10696 051716 005004
10697 051720 012702 003100
10698 051724 012703 003110
10699 051730 012767 000051 131142
10700 051736 012767 000052 131136
10701 051744 012767 000053 131132
10702 051752 012767 000054 131126
10703 051760 172467 131114
10704 051764 174005
10705 051766 004567 000240
10706 051772 172467 131102
10707 051776 174004
10708 052000 004567 000226
10709 052004 172767 131070
10710 052010 004567 000216
10711 052014 172667 131060
10712 052020 004567 000206
10713 052024 172567 131050
10714 052030 004567 000176
10715 052034 172467 131040
10716
10717 052040 174067 131044
10718 052044 004567 000246
10719 052050 001403
10720 052052 004737 140132
10721 052056 104003
10722
10723 052060 004567 000200
10724 052064 174167 131020
10725 052070 004567 000222
10726 052074 001403
10727 052076 004737 140132
10728 052102 104003
10729
10730 052104 004567 000154
10731 052110 174267 130774
10732 052114 004567 000176
10733 052120 001403
10734 052122 004737 140132
10735 052126 104003
10736
10737 052130 004567 000130
10738 052134 174367 130750
10739 052140 004567 000152
10740 052144 001403
10741 052146 004737 140132
10742 052152 104003
10743

;
;SETUP FPP ACC POINTER
;POINT TO TEST DATA
;POINT TO RECEIVED DATA
;SETUP EXPECTED DATA
;
;
;MOVE DATA TEMPORARILY
;PUT DATA INTO TEST REGISTER
;SUBTRACT TEN FROM EACH EXPECTED DATA
;MOVE DATA TEMPORARILY
;MOVE DATA INTO TEST REGISTER
;SUBTRACT 10 FROM TEST DATA WORDS
;STORE INTO TEST REGISTER
;GET NEXT SET OF UNIQUE DATA WORDS
;STORE INTO TEST REGISTER
;GET NEXT SET OF TEST DATAS
;LOAD TEST REGISTER
;GET NEXT SET OF TEST WORDS
;LOAD FINAL TEST REGISTER
;ALL REGISTER CONTAIN UNIQUE TEST WORDS
;STORE ACO,RESULT
;CHECK RESULT
;BRANCH IF GOOD
;DETERMINE FLOATING POINT FAULT. $$$
;FPP ERROR
;BAD ACO
;UPDATE EXPECTED RESULT
;STORE AC1 RESULT
;CHECK RESULT
;BRANCH IF GOOD
;DETERMINE FLOATING POINT FAULT. $$$
;FPP ERROR
;BAD RESULT AC1
;UPDATE EXPECTED RESULT
;STORE AC2 RESULT
;CHECK RESULT
;BRANCH IF GOOD
;DETERMINE FLOATING POINT RESULT. $$$
;FPP ERROR
;BAD AC2 RESULT
;UPDATE EXPECTED RESULT
;SAVE TEST RESULT
;CHECK RESULT
;BRANCH IF GOOD
;DETERMINE FLOATING POINT FAULT. $$$
;FPP ERROR
;BAD AC3 RESULT
MFA: SETD
CLR R0
CLR R4
MOV #BTEXP,R2
MOV #BTRES,R3
MOV #51,BTEXP
MOV #52,BTEXP+2
MOV #53,BTEXP+4
MOV #54,BTEXP+6
LDD BTEXP,ACO
STD ACO,AC5
JSR R5,SUBT
LDD BTEXP,ACO
STD ACO,AC4
JSR R5,SUBT
LDD BTEXP,AC3
JSR R5,SUBT
LDD BTEXP,AC2
JSR R5,SUBT
LDD BTEXP,AC1
JSR R5,SUBT
LDD BTEXP,ACO
STD ACO,BTRES
JSR R5,BFA
BEQ BFAC1
CALL #DETFPA
ERROR +3
BFAC1: JSR R5,ADD
STD AC1,BTRES
JSR R5,BFA
BEQ BFAC2
CALL #DETFPA
ERROR +3
BFAC2: JSR R5,ADD
STD AC2,BTRES
JSR R5,BFA
BEQ BFAC3
CALL #DETFPA
ERROR +3
BFAC3: JSR R5,ADD
STD AC3,BTRES
JSR R5,BFA
BEQ BFAC4
CALL #DETFPA
ERROR +3

```

FLOATING POINT TESTS

```

10744 052154 004567 000104          BFAC4: JSR    R5,ADDT          ;UPDATE EXPECTED RESULT
10745 052160 172704                LDD    AC4,AC3          ;SAVE TEMPORARY
10746 052162 174367 130722          STD    AC3,BTRES        ;STORE AC4 RESULT
10747 052166 004567 000124          JSR    R5,BFA           ;CHECK RESULT
10748 052172 001403                BEQ    BFAC5            ;BRANCH IF GOOD
10749 052174 004737 140132          CALL   @@DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10750 052200 104003                ERROR  +3              ;FPP ERROR
10751                                ;BAD AC4 RESULT
10752 052202 004567 000056          BFAC5: JSR    R5,ADDT          ;UPDATE EXPECTED RESULT
10753 052206 172605                LDD    AC5,AC2          ;SAVE TEMPORARY COPY
10754 052210 174267 130674          STD    AC2,BTRES        ;MOVE AC5 RESULT
10755 052214 004567 000076          JSR    R5,BFA           ;CHECK RESULT
10756 052220 001456                BEQ    BFAE             ;BRANCH IF GOOD
10757 052222 004737 140132          CALL   @@DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10758 052226 104003                ERROR  +3              ;FPP ERROR
10759                                ;BAD AC5 RESULT
10760 052230 000452                BR     BFAE             ;EXIT MODULE
10761
10762 052232 162767 000010 130640  SUBT:  SUB    @10,BTEXP          ;UPDATE EXPECTED CONTENTS
10763 052240 162767 000010 130634  SUB    @10,BTEXP+2        ;UPDATE EXPECTED CONTENTS
10764 052246 162767 000010 130630  SUB    @10,BTEXP+4        ;UPDATE EXPECTED CONTENTS
10765 052254 162767 000010 130624  SUB    @10,BTEXP+6        ;UPDATE EXPECTED CONTENTS
10766 052262 000205                RTS    R5                ;
10767 052264 062767 000010 130606  ADDT:  ADD    @10,BTEXP          ;UPDATE EXPECTED CONTENTS
10768 052272 062767 000010 130602  ADD    @10,BTEXP+2        ;UPDATE EXPECTED CONTENTS
10769 052300 062767 000010 130576  ADD    @10,BTEXP+4        ;UPDATE EXPECTED CONTENTS
10770 052306 062767 000010 130572  ADD    @10,BTEXP+6        ;UPDATE EXPECTED CONTENTS
10771 052314 000205                RTS    R5                ;
10772
10773 052316 026767 130556 130564  BFA:   CMP    BTEXP,BTRES        ;VERIFY CONTENTS
10774 052324 001013                BNE    BFB              ;EXIT IF NOT ZERO
10775 052326 026767 130550 130556  CMP    BTEXP+2,BTRES+2    ;VERIFY CONTENTS
10776 052334 001007                BNE    BFB              ;EXIT IF NOT ZERO
10777 052336 026767 130542 130550  CMP    BTEXP+4,BTRES+4    ;VERIFY CONTENTS
10778 052344 001003                BNE    BFB              ;EXIT IF NOT ZERO
10779 052346 026767 130534 130542  CMP    BTEXP+6,BTRES+6    ;VERIFY CONTENTS
10780 052354 000205                BFB:   RTS    R5         ;GO BACK TO CALLING ROUTINE
10781
10782
10783 052356                BFAE:
10784
10785
10786
10789 052356                TSFP1:
10790                                ; TEST LDFPS AND STFPS MODE 0
10791 052356 005037 140054          CLR    @TRPFLG          ;CLEAR TRAP FLAG
10792 052362 012704 147757          MOV    @147757,R4       ;SETUP DATA TO BE LOADED
10793 052366 004767 000032          JSR    PC,LOST          ;LOAD AND STORE FPS WITH DATA
10794 052372 012704 105252          MOV    @105252,R4       ;SETUP DATA TO BE LOADED
10795 052376 004767 000022          JSR    PC,LOST          ;LOAD AND STORE FPS WITH DATA
10796 052402 012704 042505          MOV    @42505,R4        ;SETUP DATA TO BE LOADED
10797 052406 004767 000012          JSR    PC,LOST          ;LOAD AND STORE FPS WITH DATA
10798 052412 005004                CLR    R4               ;SETUP DATA TO BE LOADED
10799 052414 004767 000004          JSR    PC,LOST          ;LOAD AND STORE FPS WITH DATA
10800
10801
10802 052420 000167 000020          JMP    FIN1

```

FLOATING POINT TESTS

```

10803
10804 052424 170104
10805 052426 170201
10806 052430 020401
10807 052432 001403
10808 052434 004737 140132
10809 052440 104003
10810
10811 052442 000207
10812 052444 000240
10813
10816 052446
10817
10818 052446 005037 140054
10819 052452 012704 000017
10820 052456 004767 000032
10821 052462 012704 000012
10822 052466 004767 000022
10823 052472 012704 000005
10824 052476 004767 000012
10825 052502 005004
10826 052504 004767 000004
10827
10828 052510 000167 000030
10829
10830 052514 170104
10831 052516 170000
10832 052520 013701 177776
10833 052524 042701 177760
10834 052530 020401
10835
10836 052532 001403
10837 052534 004737 140132
10838 052540 104003
10839
10840 052542 000207
10841 052544
10842
10845 052544
10846
10847 052544 005037 140054
10848 052550 012704 000200
10849 052554 170104
10850 052556 170001
10851 052560 170201
10852 052562 020127 000000
10853 052566 001403
10854 052570 004737 140132
10855 052574 104003
10856
10857 052576 170011
10858 052600 170201
10859 052602 020104
10860 052604 001403
10861 052606 004737 140132
10862 052612 104003
10863

; LOST: LDFPS R4 ;LOAD FPS WITH DATA
STFPS R1 ;LOAD R1 WITH (FPS)
CMP R4,R1 ;DID THE INSTRUCTIONS WORK
BEQ 1$ ;YES GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR

1$: RTS PC ;RETURN
FIN1: NOP

; TSFP2:
; TEST CFCC
CLR @TRPFLG ;CLEAR TRAP FLAG
MOV @17,R4 ;SETUP DATA TO BE LOADED
JSR PC,TSF2 ;LOAD FPS AND COPY CONDITION CODES TO PS
MOV @12,R4 ;SETUP DATA TO BE LOADED
JSR PC,TSF2 ;LOAD FPS AND COPY CONDITION CODES TO PS
MOV @5,R4 ;SETUP DATA TO BE LOADED
JSR PC,TSF2 ;LOAD FPS AND COPY CONDITION CODES TO PS
CLR R4 ;SETUP DATA TO BE LOADED
JSR PC,TSF2 ;LOAD FPS AND COPY CONDITION CODES TO PS

;
; JMP FIN2

; TSF2: LDFPS R4 ;LOAD FPS
CFCC ;COPY CONDITION CODES TO PS
MOV @177776,R1 ;SAVE PS TO R1
BIC @177760,R1 ;MASK OUT UNWANTED BITS
CMP R4,R1 ;WAS CONDITION CODE BITS TRANSFERRED
;CORRECTLY
BEQ 1$ ;YES GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR

1$: RTS PC ;RETURN
FIN2:

; TSFP3:
; TEST SETF, SETD, SETI, SETL
CLR @TRPFLG ;CLEAR TRAP FLAG
MOV @200,R4 ;SETUP DATA TO BE LOADED
LDFPS R4 ;LOAD FPS
SETF ;MAKE FD=0
STFPS R1 ;STORE FPS
CMP R1,#0 ;IS FD=0
BEQ 1$ ;YES GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR

1$: SETD ;MAKE FD=1
STFPS R1 ;STORE FPS
CMP R1,R4 ;IS FD=1
BEQ 2$ ;YES GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR
    
```

FLOATING POINT TESTS

```

10864 052614 012704 000100      2:  MOV    #100,R4          ;SETUP DATA TO BE LOADED
10865 052620 170104             LDFPS  R4              ;LOAD FPS
10866 052622 170002             SETI   ;MAKE FL=0
10867 052624 170201             STFPS  R1              ;STORE FPS
10868 052626 020127 000000      CMP    R1,#0           ;IS FL=0
10869 052632 001403             BEQ    3:              ;YES GO ON
10870 052634 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT $$$
10871 052640 104003             ERROR  +3              ;FPP ERROR
10872                               ;NO GO TO ERROR
10873 052642 170012             3:  SETL   ;MAKE FL=1
10874 052644 170201             STFPS  R1              ;STORE FPS
10875 052646 020104             CMP    R1,R4          ;IS FL=1
10876 052650 001403             BEQ    4:              ;YES GO ON
10877 052652 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
10878 052656 104003             ERROR  +3              ;FPP ERROR
10879                               ;NO GO TO ERROR
10880 052660             4:  ;
10881                               ;
10884 052660             ;TSFP4:
10885                               ;
10886 052660 005037 140054      ; TEST ILLEGAL OP CODES AND STST
10887 052664 012705 170003      CLR    @TRPFLG         ;CLEAR TRAP FLAG
10888 052670 013746 000244      MOV    #170003,R5      ;INIT OP CODE
10889 052674 012737 053030 000244  MOV    @244,-(SP)      ;SAVE FP VECTOR
10890 052702 013746 000004      MOV    @ILLOP1,@244   ;SETUP NEW VECTOR
10891 052706 012737 053114 000004  MOV    @4,-(SP)       ;SAVE TIME OUT VECTOR
10892 052714 013746 000010      MOV    @TIMEOU,@4     ;SETUP NEW VECTOR
10893 052720 012737 053124 000010  MOV    @10,-(SP)     ;SAVE ILLEGAL VECTOR
10894 052726 005003             MOV    @ILLOP2,@10    ;SETUP NEW VECTOR
10895 052730 170103             D1:  CLR    R3            ;
10896 052732 005002             LDFPS  R3            ;CLEAR FPS
10897 052734 010537 052740      CLR    R2            ;
10898 052740 000000             MOV    R5,@02        ;SETUP THE ILLEGAL INST
10899 052742 170000             D2:  .WORD 0          ;
10900 052744 005202             D3:  CFCC            ;MEMORY WORDS TO BE USED WITH
10901 052746 005202             INC    R2            ;EXECUTION OF ILLEGAL OP CODE
10902 052750 170201             INC    R2            ;
10903 052752 004737 140132      STFPS  R1            ;SAVE FPS
10904 052756 104003             CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
10905                               ERROR  +3              ;FPP ERROR
10906 052760 022705 170010      D4:  CMP    @170010,R5 ;GO TO ERROR
10907 052764 001003             BNE    D5            ;COMPUTE NEXT OP CODE
10908 052766 012705 170013      MOV    @170013,R5     ;
10909 052772 000755             BR     D1            ;
10910 052774 022705 170077      D5:  CMP    @170077,R5 ;
10911 053000 001001             BNE    D6            ;
10912 053002 000402             BR     D7            ;
10913 053004 005205             D6:  INC    R5        ;
10914 053006 000747             BR     D1            ;
10915 053010 012637 000010      D7:  MOV    (SP)+,@10  ;RESTORE VECTORS
10916 053014 012637 000004      MOV    (SP)+,@4      ;
10917 053020 012637 000244      MOV    (SP)+,@244    ;
10918                               ;
10919                               ;
10920 053024 000167 000104      ; JMP    FIN4         ;
10921                               ;
10922 053030 022716 052742      ;ILLOP1: CMP    @03,(SP) ;DID TRAP OCCUR ON TEST INST

```

FLOATING POINT TESTS

```

10923 053034 001403          BEQ      1#          ;YES GO ON
10924 053036 004737 140132  CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10925 053042 104003          ERROR    +3          ;FPP ERROR
10926                                ;NO GO TO ERROR
10927 053044 022626          1#:    CMP      (SP)+,(SP)+ ;CLEAN UP STACK
10928 053046 170201          STFPS   R1          ;STORE FPS
10929 053050 022701 100000  CMP      @100000,R1 ;IS FPS CORRECT
10930 053054 001403          BEQ      2#          ;YES GO ON
10931 053056 004737 140132  CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10932 053062 104003          ERROR    +3          ;FPP ERROR
10933                                ;NO GO TO ERROR
10934 053064 005004          2#:    CLR      R4          ;INT R4 TO A KNOWN STATE
10935 053066 170304          STST   R4          ;STORE FEC AT R4
10936                                ;IF THE DESTINATION MODE IS IMPROPERLY
10937                                ;DECODED AN ODD ADDRESS TRAP TO 4
10938                                ;SHOULD OCCUR
10939 053070 022704 000002  CMP      @2,R4      ;IS FEC CORRECT
10940 053074 001002          BNE     3#          ;NO GO TO ERROR
10941 053076 000167 177656  JMP      D4          ;YES GO ON
10942 053102 004737 140132  3#:    CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10943 053106 104003          ERROR    +3          ;FPP ERROR
10944                                ;GO TO ERROR
10945 053110 000167 177644  JMP      D4          ;THEN GO ON
10946                                ;
10947 053114 004737 140132  ;TIMEOU: CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10948 053120 104003          ERROR    +3          ;FPP ERROR
10949                                ;ERROR BECAUSE OF TRAP TO 4
10950 053122 000006          RTT                                ;RETURN
10951                                ;
10952 053124 004737 140132  ;ILLOP2: CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT $$$
10953 053130 104003          ERROR    +3          ;FPP ERROR
10954                                ;ERROR BECAUSE OF TRAP TO 10
10955 053132 000006          RTT                                ;RETURN
10956 053134          FIN4:
10957                                ;
10960 053134          ;TSFPS:
10961                                ;
10962 053134 005037 140054          ; TEST FID (INTERRUPT DISABLE BIT)
10963 053140 013746 000244  CLR      @TRPFLG    ;CLEAR TRAP FLAG
10964 053144 012737 053230 000244  MOV      @244,-(SP) ;SAVE FP VECTOR
10965 053152 012703 040000          MOV      @ILL,@244  ;SETUP NEW VECTOR
10966 053156 170103          MOV      @0000,R3   ;SETUP DATA TO BE LOADED
10967 053160 170020          LDFPS   R3          ;LOAD FPS, FID=1
10968 053162 170000          .WORD   170020     ;ILLEGAL FP INSTRUCTION
10969 053164 170201          CFCC                                ;
10970 053166 022701 140000  STFPS   R1          ;SEE IF ERROR WAS RECORDED IN FPS
10971 053172 001403          CMP      @140000,R1 ;
10972 053174 004737 140132  BEQ      1#          ;YES GO ON
10973 053200 104003          CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10974          ERROR    +3          ;FPP ERROR
10975          ;NO GO TO ERROR
10975 053202 170304          1#:    STST   R4          ;SEE IF FEC=2
10976 053204 022704 000002  CMP      @2,R4      ;
10977 053210 001403          BEQ      2#          ;YES GO ON
10978 053212 004737 140132  CALL     @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
10979 053216 104003          ERROR    +3          ;FPP ERROR
10980          ;NO GO TO ERROR
10981 053220 012637 000244  2#:    MOV      (SP)+,@244 ;RESTORE VECTOR

```

FLOATING POINT TESTS

```

10982
10983 053224 000167 000010      ;      JMP      FIN5
10984
10985 053230 004737 140132      ; ILL:  CALL    @@DETFPA      ; DETERMINE FLOATING POINT FAULT. $$$
10986 053234 104003              ;      ERROR   +3              ; FPP ERROR
10987                                ;                                ; FID ERROR
10988 053236 000006              ;      RTT                ; RETURN
10989 053240
10990
10993 053240
10994
10995 053240 005037 140054      ;      TEST LDD, STD FSRC AND FDST MODE 1
10996 053244 005004              ;      CLR     @@TRPFLG      ; CLEAR TRAP FLAG
10997 053246 170104              ;      CLR     R4            ; SETUP TO LOAD DATA
10998 053250 170011              ;      LDFPS  R4            ; CLEAR FPS
10999 053252 013746 000004      ;      SETD   ; SET FD TO 1
11000 053256 012737 053430 000004 ;      MOV     @@4, -(SP)    ; SAVE TIMEOUT VECTOR
11001 053264 012704 053420      ;      MOV     @TSF6, @@4    ; SETUP NEW VECTOR
11002 053270 172414              ;      MOV     @TS6DAT, R4   ; SETUP POINTER TO DATA
11003 053272 020427 053420      ;      LDD    (R4), ACO     ; TEST INSTRUCTION
11004 053276 001403              ;      CMP    R4, @TS6DAT   ; IS R4 CORRECT
11005 053300 004737 140132      ;      BEQ    1$            ; YES GO ON
11006 053304 104003              ;      CALL   @@DETFPA      ; DETERMINE FLOATING POINT FAULT. $$$
11007                                ;      ERROR   +3              ; FPP ERROR
11008 053306 012701 053410      ;      1$:  MOV     @TS6DA, R1 ; NO GO TO ERROR
11009 053312 012703 000004      ;      MOV     @4, R3        ; SETUP POINTER TO DATA
11010 053316 022421              ;      2$:  CMP    (R4)+, (R1)+ ; INIT COUNTER
11011 053320 001403              ;      BEQ    3$            ; WAS SOURCE DATA ALTERED
11012 053322 004737 140132      ;      CALL   @@DETFPA      ; NO GO ON
11013 053326 104003              ;      ERROR   +3              ; DETERMINE FLOATING POINT FAULT. $$$
11014                                ;                                ; FPP ERROR
11015 053330 077306              ;      3$:  SOB    R3, 2$    ; YES GO TO ERROR
11016 053332 012704 003162      ;      MOV     @TSTLOC, R4   ; ARE WE DONE
11017 053336 174014              ;      STD    ACO, (R4)     ; SETUP POINTER FOR DATA
11018 053340 020427 003162      ;      CMP    R4, @TSTLOC   ; TEST INSTRUCTION
11019 053344 001403              ;      BEQ    4$            ; IS R4 CORRECT
11020 053346 004737 140132      ;      CALL   @@DETFPA      ; YES GO ON
11021 053352 104003              ;      ERROR   +3              ; DETERMINE FLOATING POINT FAULT. $$$
11022                                ;                                ; FPP ERROR
11023 053354 012701 053410      ;      4$:  MOV     @TS6DA, R1 ; NO GO TO ERROR
11024 053360 012703 000004      ;      MOV     @4, R3        ; SETUP POINTER TO DATA
11025 053364 022421              ;      5$:  CMP    (R4)+, (R1)+ ; INIT COUNTER
11026 053366 001403              ;      BEQ    6$            ; IS DESTINATION DATA CORRECT
11027 053370 004737 140132      ;      CALL   @@DETFPA      ; YES GO ON
11028 053374 104003              ;      ERROR   +3              ; DETERMINE FLOATING POINT FAULT. $$$
11029                                ;                                ; FPP ERROR
11030 053376 077306              ;      6$:  SOB    R3, 5$    ; NO GO TO ERROR
11031 053400 012637 000004      ;      MOV     (SP)+, @@4    ; ARE WE DONE
11032                                ;                                ; RESTORE VECTOR
11033
11034 053404 000167 000030      ;      JMP      FIN6
11035
11036 053410 177777              ; TS6DA: .WORD 177777
11037 053412 000000              ;      .WORD 000000
11038 053414 052525              ;      .WORD 052525
11039 053416 125252              ;      .WORD 125252
11040 053420 177777              ; TS6DAT: .WORD 177777

```


FLOATING POINT TESTS

```

11041 053422 000000 .WORD 000000
11042 053424 052525 .WORD 052525
11043 053426 125252 .WORD 125252
11044
11045 053430 004737 140132 ;
;TSF6: CALL @@DETFPA ; DETERMINE FLOATING POINT FAULT. $$$
11046 053434 104003 ERROR +3 ;FPP ERROR
11047 ; ;ODD ADDRESS TRAP
11048 053436 000006 RTT ;RETURN
11049 053440
11050
11053 053440
11054 ;
;TSFP7:
; TEST LDD, LDF FSRC MODE 0
11055 053440 005037 140054 CLR @@TRPFLG ;CLEAR TRA FLAG
11056 053444 012704 000200 MOV @200,R4 ;SETUP TO LOAD FPS
11057 053450 170104 LDFPS R4 ;LOAD FPS, FD=1
11058 053452 013746 000004 MOV @@4,-(SP) ;SAVE TIMEOUT VECTOR
11059 053456 012737 053624 000004 MOV @TSF7,@@4 ;SETUP NEW VECTOR
11060 053464 012704 053634 MOV @TS7DA1,R4 ;SETUP POINTER TO DATA
11061 053470 172414 LDD (R4),ACO ;CLEAR ACO
11062 053472 012701 053644 MOV @TS7DA2,R1 ;SETUP POINTER TO DATA
11063 053476 172511 LDD (R1),AC1 ;LOAD AC1 WITH DATA
11064 053500 172401 LDD AC1,ACO ; TEST INSTRUCTION
11065 053502 012704 003162 MOV @TSTLOC,R4 ;
11066 053506 174114 STD AC1,(R4) ;CHECK IF AC1 HAS BEEN ALTERED
11067 053510 004767 000072 JSR PC,CHECK7 ;
11068 053514 012704 003162 MOV @TSTLOC,R4 ;
11069 053520 012701 053644 MOV @TS7DA2,R1 ; ;SETUP POINTERS FOR DATA
11070 053524 174014 STD ACO,(R4) ;CHECK IF ACO RECEIVED CORRECT DATA
11071 053526 004767 000054 JSR PC,CHECK7 ;
11072 053532 012701 053634 MOV @TS7DA1,R1 ;SETUP POINTER TO DATA
11073 053536 172511 LDD (R1),AC1 ;CLEAR AC1
11074 053540 170001 SETF ;SET FD=0
11075 053542 172401 LDF AC1,ACO ; TEST INSTRUCTION
11076 053544 170011 SETD ;SET FD=1
11077 053546 012704 003162 MOV @TSTLOC,R4 ;SETUP POINTER TO DATA
11078 053552 174114 STD AC1,(R4) ;CHECK IF AC1 HAS BEEN ALTERED
11079 053554 004767 000026 JSR PC,CHECK7 ;
11080 053560 012704 053654 MOV @TS7DA4,R4 ;SETUP POINTERS FOR DATA
11081 053564 012701 003162 MOV @TSTLOC,R1 ;
11082 053570 174011 STD ACO,(R1) ;CHECK IF ACO HAS CORRECT DATA
11083 053572 004767 000010 JSR PC,CHECK7 ;
11084 053576 012637 000004 MOV (SP)+,@@4 ;RESTORE VECTOR
11085
11086 ;
11087 053602 000167 000056 JMP FIN7
11088 ;
;CHECK7: MOV @4,R3 ;INIT COUNTER
11089 053606 012703 000004 CHECK7: CMP (R4)+,(R1)+ ;IS DATA OK
11090 053612 022421 BEQ CHK7 ;YES GO ON
11091 053614 001401 ERROR +3 ;FPP ERROR
11092 053616 104003 ;NO GO TO ERROR
11093
11094 053620 077304 CHK7: SOB R3,CHK7 ;ARE WE DONE
11095 053622 000207 RTS PC ;YES RETURN
11096
11097 053624 004737 140132 ;TSF7: CALL @@DETFPA ; DETERMINE FLOATING POINT FAULT. $$$
11098 053630 104003 ERROR +3 ;FPP ERROR
11099 ;ODD ADDRESS TRAP

```

FLOATING POINT TESTS

11100 053632 000006		RTT			
11101					;RETURN
11102 053634 000000		TS7DA1:	.WORD 0		
11103 053636 000000			.WORD 0		
11104 053640 000000			.WORD 0		
11105 053642 000000			.WORD 0		
11106 053644 037641		TS7DA2:	.WORD 37641		
11107 053646 065121			.WORD 65121		
11108 053650 037373			.WORD 37373		
11109 053652 022265			.WORD 22265		
11110 053654 000000		TS7DA4:	.WORD 0		
11111 053656 000000			.WORD 0		
11112 053660 037373			.WORD 37373		
11113 053662 022265			.WORD 22265		
11114 053664		FIN7:			
11115					
11118 053664		TSFP10:			
11119					
11120 053664 005037 140054			TEST STD, STF FDST MODE 0		
11121 053670 012704 000200			CLR @TRPFLG		;CLEAR TRAP FLAG
11122 053674 170104			MOV @200,R4		;SETUP TO LOAD FPS
11123 053676 013746 000004			LDFPS R4		;LOAD FPS, FD=1
11124 053702 012737 054054 000004			MOV @@4,-(SP)		;SAVE TIMEOUT VECTOR
11125 053710 012704 054064			MOV @TSF10,@@4		;SETUP NEW VECTOR
11126 053714 172414			MOV @TS10D1,R4		;SETUP POINTER TO DATA
11127 053716 012701 054074			LDD (R4),ACO		;CLEAR ACO
11128 053722 172511			MOV @TS10D2,R1		;SETUP POINTER TO DATA
11129 053724 174100			LDD (R1),AC1		;LOAD AC1 WITH DATA
11130 053726 012704 003162			STD AC1,ACO		; TEST INSTRUCTION
11131 053732 174114			MOV @TSTLOC,R4		
11132 053734 004767 000072			STD AC1,(R4)		;CHECK IF AC1 HAS BEEN ALTERED
11133 053740 012704 003162			JSR PC,CHEC10		
11134 053744 012701 054074			MOV @TSTLOC,R4		;SETUP POINTERS FOR DATA
11135 053750 174014			MOV @TS10D2,R1		
11136 053752 004767 000054			STD ACO,(R4)		;CHECK IF ACO RECEIVED CORRECT DATA
11137 053756 012701 054064			JSR PC,CHEC10		
11138 053762 172511			MOV @TS10D1,R1		;SETUP POINTER TO DATA
11139 053764 170001			LDD (R1),AC1		;CLEAR AC1
11140 053766 174100			SETF		;SET FD=0
11141 053770 170011			STF AC1,ACO		; TEST INSTRUCTION
11142 053772 012704 003162			SETD		;SET FD=1
11143 053776 174114			MOV @TSTLOC,R4		;SETUP POINTER TO DATA
11144 054000 004767 000026			STD AC1,(R4)		;CHECK IF AC1 HAS BEEN ALTERED
11145 054004 012704 054104			JSR PC,CHEC10		
11146 054010 012701 003162			MOV @TS10D4,R4		;SETUP POINTERS FOR DATA
11147 054014 174011			MOV @TSTLOC,R1		
11148 054016 004767 000010			STD ACO,(R1)		;CHECK IF ACO HAS CORRECT DATA
11149 054022 012637 000004			JSR PC,CHEC10		
11150			MOV (SP)+,@@4		;RESTORE VECTOR
11151					
11152 054026 000167 000062			JMP FIN10		
11153					
11154 054032 012703 000004		CHEC10:	MOV @4,R3		;INIT COUNTER
11155 054036 022421		CH10:	CMP (R4)+,(R1)+		;IS DATA OK
11156 054040 001403			BEQ CHK10		;YES GO ON
11157 054042 004737 140132			CALL @@DETFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
11158 054046 104003			ERROR +3		;FPP ERROR

FLOATING POINT TESTS

```

11159
11160 054050 077306
11161 054052 000207
11162
11163 054054 004737 140132
11164 054060 104003
11165
11166 054062 000006
11167
11168 054064 000000
11169 054066 000000
11170 054070 000000
11171 054072 000000
11172 054074 177777
11173 054076 111236
11174 054100 100045
11175 054102 003651
11176 054104 000000
11177 054106 000000
11178 054110 100045
11179 054112 003651
11180 054114
11181
11184 054114
11185
11186 054114 005037 140054
11187 054120 012704 000200
11188 054124 170104
11189 054126 012704 054210
11190 054132 172414
11191 054134 170400
11192 054136 170203
11193 054140 012704 003162
11194 054144 174014
11195 054146 012701 000004
11196 054152 022427 000000
11197 054156 001403
11198 054160 004737 140132
11199 054164 104003
11200
11201 054166 077107
11202 054170 020327 000204
11203 054174 001403
11204 054176 004737 140132
11205 054202 104003
11206
11207 054204
11208
11209 054204 000167 000010
11210
11211 054210 177777
11212 054212 177777
11213 054214 177777
11214 054216 177777
11215 054220
11216
11219 054220

CHK10: SOB R3,CH10 ;NO GO TO ERROR
RTS PC ;ARE WE DONE
;YES RETURN

;
TSF10: CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;ODD ADDRESS TRAP
RTT ;

;
TS10D1: .WORD 0
.WORD 0
.WORD 0
.WORD 0
TS10D2: .WORD 177777
.WORD 111236
.WORD 100045
.WORD 3651
TS10D4: .WORD 0
.WORD 0
.WORD 100045
.WORD 3651

FIN10:
;
TSFP11:
; TEST FDST SINGLE OPERAND MODE 0
CLR @TRPFLG ;CLEAR TRAP FLAG
MOV #200,R4 ;SETUP TO LOAD FPS
LDFPS R4 ;SET FD=1
MOV @TS11D1,R4 ;SETUP POINTER TO DATA
LDD (R4),ACO ;LOAD ALL ONES TO ACO
CLRD ACO ; TEST INSTRUCTION
STFPS R3 ;GET FPS
;
STD ACO,(R4) ;CHECK ACO FOR ALL ZEROES
MOV #4,R1 ;INIT COUNTER
1$: CMP (R4)+, #0
BEQ 2$ ;OK GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR
2$: SOB R1,1$ ;ARE WE DONE
CMP R3,#204 ;CHECK FPS
BEQ 3$ ;OK GO ON
CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;NO GO TO ERROR
3$:
;
JMP FIN11

;
TS11D1: .WORD 177777
.WORD 177777
.WORD 177777
.WORD 177777

FIN11:
;
TSFP12:

```

FLOATING POINT TESTS

```

11220      ;      TEST FDST SOP MODE 0 WITH ILLEGAL AC7
11221 054220 005037 140054      CLR      @TRPFLG      ;CLEAR TRAP FLAG
11222 054224 012703 040200      MOV      @40200,R3    ;SETUP TO LOAD FPS
11223 054230 170103              LDFPS   R3            ;SET FID=1, AND FD=1
11224 054232 170407              CLRD   AC7           ; TEST INSTRUCTION
11225 054234 170204              STFPS  R4            ;GET FPS
11226 054236 170305              STST   R5            ;GET FEC
11227 054240 022704 140200      CMP     @140200,R4    ;IS FPS CORRECT
11228 054244 001403              BEQ    1$           ;YES GO ON
11229 054246 004737 140132      CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11230 054252 104003              ERROR  +3          ;FPP ERROR
11231              ;NO GO TO ERROR
11232 054254 022705 000002      1$:    CMP     @2,R5    ;IS FEC CORRECT
11233 054260 001403              BEQ    2$           ;YES GO ON
11234 054262 004737 140132      CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11235 054266 104003              ERROR  +3          ;FPP ERROR
11236              ;NO GO TO ERROR
11237 054270              2$:
11238              ;
11241 054270              ;TSFP13:
11242              ;
11243 054270 013716 000004      ;      TEST FDST SOP MODE 1
11244 054274 012737 054424 000004      MOV     @4,-(SP)     ;SAVE TIMEOUT VECTOR
11245 054302 005037 140054      MOV     @TSF13,@4    ;SETUP NEW VECTOR
11246 054306 012702 000200      CLR     @TRPFLG     ;CLEAR TRAP FLAG
11247 054312 170102              MOV     @200,R2     ;SETUP TO LOAD FPS
11248 054314 012705 000004      LDFPS  R2           ;SET FD=1
11249 054320 012704 003162      MOV     @4,R5       ;INIT COUNTER
11250 054324 012724 177777      MOV     @TSTLOC,R4  ;SETUP PCINTER TO TEST LOCATION
11251 054330 071503              100$:  MOV     @177777,(R4)+ ;MOVE ALL ONES TO TEST LOCATION
11252 054332 012702 003162      SOB    R5,100$     ;ARE WE DONE
11253 054336 170412              MOV     @TSTLOC,R2  ;SETUP POINTER TO DATA
11254 054340 170203              CLRD   (R2)         ; TEST INSTRUCTION
11255 054342 020227 003162      STFPS  R3           ;GET FPS
11256 054346 001403              CMP     R2,@TSTLOC  ;WAS R2 ALTERED
11257 054350 004737 140132      BEQ    1$           ;NO GO ON
11258 054354 104003              CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11259              ERROR  +3          ;FPP ERROR
11260 054356 012701 000004      1$:    MOV     @4,R1       ;YES GO TO ERROR
11261 054362 022227 000000      2$:    CMP     (R2)+,@0  ;INIT COUNTER
11262 054366 001403              BEQ    3$           ;CHECK LOCATION FOR 0
11263 054370 004737 140132      CALL   @DETFPA      ;OK GO ON
11264 054374 104003              ERROR  +3          ;DETERMINE FLOATING POINT FAULT. $$$
11265              ;FPP ERROR
11266 054376 077107              ;NO GO TO ERROR
11267 054400 020327 000204      3$:    SOB    R1,2$     ;ARE WE DONE
11268 054404 001403              CMP     R3,@204     ;CHECK FPS
11269 054406 004737 140132      BEQ    4$           ;OK GO ON
11270 054412 104003              CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11271              ERROR  +3          ;FPP ERROR
11272 054414 012637 000004      4$:    MOV     (SP)+,@4  ;NO GO TO ERROR
11273              ;RESTORE VECTOR
11274              ;
11275 054420 000167 000010      ;      JMP     FIN13
11276              ;
11277 054424 004737 140132      ;TSF13: CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11278 054430 104003              ERROR  +3          ;FPP ERROR

```

FLOATING POINT TESTS

```

11279                                     ;ODD ADDRESS TRAP
11280 054432 000006                       RTT                               ;RETURN
11281                                     ;
11282 054434                               ;FIN13:
11283                                     ;
11286 054434                               ;TSFP14:
11287                                     ;
11288 054434 013746 000004                 ; TEST FDST SOP MODE 2
11289 054440 012737 054574 000004         MOV    @@4,-(SP)                    ;SAVE TIMEOUT VECTOR
11290 054446 005037 140054                 MOV    @TSF14,@@4                  ;SETUP NEW VECTOR
11291 054452 012702 000200                 CLR    @TRPFLG                     ;CLEAR TRAP FLAG
11292 054456 170102                         MOV    @200,R2                     ;SETUP TO LOAD FPS
11293 054460 012705 000004                 LDFPS R2                           ;SET FD=1
11294 054464 012704 003162                 MOV    @4,R5                       ;INIT COUNTER
11295 054470 012724 177777                 MOV    @TSTLOC,R4                  ;SETUP POINTER TO TEST LOCATION
11296 054474 077503                         SOB    R5,100$                     ;MOVE ALL ONES TO TEST LOCATION
11297 054476 012702 003162                 MOV    @TSTLOC,R2                  ;ARE WE DONE
11298 054502 170422                         CLRD  (R2)+                          ;SETUP POINTER TO DATA
11299 054504 170203                         STFPS R3                            ; TEST INSTRUCTION
11300 054506 020227 003172                 CMP    R2,@TSTLOC+10               ;GET FPS
11301 054512 001403                         BEQ    1$                            ;IS R2 CORRECT
11302 054514 004737 140132                 CALL   @@DETFPA                     ;YES GO ON
11303 054520 104003                         ERROR  +3                            ;DETERMINE FLOATING POINT FAULT. $$$
11304                                     ;FPP ERROR
11305 054522 012702 003162                 1$:  MOV    @TSTLOC,R2                ;NO GO TO ERROR
11306 054526 012701 000004                 MOV    @4,R1                        ;SETUP POINTER TO DATA
11307 054532 022227 000000                 2$:  CMP    (R2)+,@0                 ;INIT COUNTER
11308 054536 001403                         BEQ    3$                            ;CHECK LOCATION FOR 0
11309 054540 004737 140132                 CALL   @@DETFPA                     ;YES GO ON
11310 054544 104003                         ERROR  +3                            ;DETERMINE FLOATING POINT FAULT. $$$
11311                                     ;FPP ERROR
11312 054546 077107                         3$:  SOB    R1,2$                     ;NO GO TO ERROR
11313 054550 020327 000204                 CMP    R3,@204                      ;ARE WE DONE
11314 054554 001403                         BEQ    4$                            ;CHECK FPS
11315 054556 004737 140132                 CALL   @@DETFPA                     ;OK GO ON
11316 054562 104003                         ERROR  +3                            ;DETERMINE FLOATING POINT FAULT. $$$
11317                                     ;FPP ERROR
11318 054564 012637 000004                 4$:  MOV    (SP)+,@@4                ;NO GO TO ERROR
11319                                     ;RESTORE VECTOR
11320                                     ;
11321 054570 000167 000010                 ;
11322                                     ;
11323 054574 004737 140132                 ;TSFP14: CALL @@DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11324 054600 104003                         ERROR  +3                            ;FPP ERROR
11325                                     ;ODD ADDRESS TRAP
11326 054602 000006                       RTT                               ;RETURN
11327                                     ;
11328 054604 000240                               ;FIN14: NOP
11329                                     ;
11330                                     ;
11331                                     ;
11332                                     ;
11333                                     ;
11334                                     ;-----
11335                                     ;TEST FDST SOP MODE 3
11336                                     ;
11337 054606                               ;TSFP15:
11338                                     ;
11339 054606 013746 000004                 ;
11339 054606 013746 000004                 MOV    @@4,-(SP)                    ;SAVE TIMEOUT VECTOR

```


FLOATING POINT TESTS

```

11399
11400 055024      ;
11401            ;TSFP16:
11402 055024 013746 000004      ;
11403 055030 012737 055202 000004      MOV    @4, (SP)          ;SAVE TIMEOUT VECTOR
11404 055036 005037 140054      MOV    @TSF16,@4        ;SETUP NEW VECTOR
11405 055042 012702 020000      CLR    @TRPFLG          ;CLEAR TRAP FLAG
11406 055046 170102 000000      MOV    @200,R2         ;SETUP TO LOAD FPS
11407 055050 012705 000010      LDFPS R2               ;SET FD=1
11408 055054 012704 003162      MOV    @8,R5           ;INIT COUNTER
11409 055060 012724 177777      MOV    @TSTLOC,R4      ;SETUP POINTER TO TEST LOCATION
11410 055064 077503 000000      MOV    @177777,(R4)+   ;INIT TEST LOCATION
11411 055066 012702 003172      SOB    R5,100$        ;ARE WE DONE
11412 055072 170442 000000      MOV    @TSTLOC+10,R2   ;SETUP POINTER TO DATA
11413 055074 170203 000000      CLRD  -(R2)           ; TEST INSTRUCTION
11414 055076 020227 003162      STFPS R3              ;GET FPS
11415 055102 001403 000000      CMP    R2,@TSTLOC      ;IS R2 CORRECT
11416 055104 004737 140132      BEQ   1$              ;YES GO ON
11417 055110 104003 000000      CALL  @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
11418            ERROR   +3      ;FPP ERROR
11419 055112 012701 000004      1$: MOV    @4,R1        ;NO GO TO ERROR
11420 055116 022227 000000      2$: CMP    (R2)+,@0     ;INIT COUNTER
11421 055122 001403 000000      BEQ   3$              ;IS LOCATION 0
11422 055124 004737 140132      CALL  @DETFPA          ;YES GO ON
11423 055130 104003 000000      ERROR +3             ;DETERMINE FLOATING POINT FAULT. $$$
11424            ;FPP ERROR
11425 055132 077107 000000      3$: SOB    R1,2$        ;NO GO TO ERROR
11426 055134 012701 000004      MOV    @4,R1          ;ARE WE DONE
11427 055140 022227 177777      4$: CMP    (R2)+,@177777 ;INIT COUNTER
11428 055144 001403 000000      BEQ   5$              ;IS LOCATION UNCHANGED
11429 055146 004737 140132      CALL  @DETFPA          ;YES GO ON
11430 055152 104003 000000      ERROR +3             ;DETERMINE FLOATING POINT FAULT. $$$
11431            ;FPP ERROR
11432 055154 077107 000000      5$: SOB    R1,4$        ;NO GO TO ERROR
11433 055156 020327 000204      CMP    R3,@204        ;ARE WE DONE
11434 055162 001403 000000      BEQ   6$              ;CHECK FPS
11435 055164 004737 140132      CALL  @DETFPA          ;OK GO ON
11436 055170 104003 000000      ERROR +3             ;DETERMINE FLOATING POINT FAULT. $$$
11437            ;FPP ERROR
11438 055172 012637 000004      6$: MOV    (SP)+,@4     ;NO GO TO ERROR
11439 055176 000167 000010      JMP   FIN16           ;RESTORE VECTOR
11440            ;
11441 055202 004737 140132      ;TSF16: CALL @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
11442 055206 104003 000000      ERROR +3             ;FPP ERROR
11443            ;
11444 055210 000006 000000      RTT                  ;ODD ADDRESS TRAP
11445            ;
11446 055212 000240 000000      FIN16: NOP           ;RETURN
11447            ;
11450            ;
11451            ;-----
11452            ;TEST FDST SOP MODE 5
11453            ;
11454            ;
11455 055214      ;TSFP17:
11456            ;
11457 055214 013746 000004      MOV    @4,-(SP)       ;SAVE TIMEOUT VECTOR

```


FLOATING POINT TESTS

```

11517
11518 055426 005037 140054          ;
11519 055432 013746 000004          CLR      @TRPFLG          ;CLEAR TRAP FLAG
11520 055436 012737 055612 000004  MOV      @4,-(SP)        ;SAVE TIMEOUT VECTOR
11521 055444 012702 000200          MOV      @TSF20,@4      ;SETUP NEW VECTOR
11522 055450 170102          MOV      @200,R2        ;SETUP TO LOAD FPS
11523 055452 012705 000010          LDFPS   R2              ;SET FD=1
11524 055456 012704 003162          MOV      @8.,R5         ;INIT COUNTER
11525 055462 012724 177777          MOV      @TSTLOC,R4     ;SETUP POINTER TO TEST LOCATION
11526 055466 077503          MOV      @177777,(R4)+  ;INIT TEST LOCATION
11527 055470 012702 003163          SOB     R5,100$        ;ARE WE DONE
11528 055474 170462 000007          MOV      @TSTLOC+1,R2   ;SETUP POINTER TO DATA
11529 055500 170203          CLRD    7(R2)          ; TEST INSTRUCTION
11530 055502 020227 003163          STFPS  R3              ;GET FPS
11531 055506 001403          CMP     R2,@TSTLOC+1    ;IS R2 CORRECT
11532 055510 004737 140132          BEQ     1$             ;YES GO ON
11533 055514 104003          CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11534                                ERROR   +3             ;FPP ERROR
11535 055516 012702 003162          ;NO GO TO ERROR
11536 055522 012701 000004          MOV      @TSTLOC,R2     ;SETUP POINTER TO DATA
11537 055526 022227 177777          MOV      @4,R1         ;INIT COUNTER
11538 055532 001403          CMP     (R2)+,@177777  ;IS DATA CORRECT
11539 055534 004737 140132          BEQ     3$             ;YES GO ON
11540 055540 104003          CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11541                                ERROR   +3             ;FPP ERROR
11542 055542 077107          ;NO GO TO ERROR
11543 055544 012701 000004          SOB     R1,2$          ;ARE WE DONE
11544 055550 022227 000000          MOV      @4,R1         ;INIT COUNTER
11545 055554 001403          CMP     (R2)+,@0       ;IS DATA CORRECT
11546 055556 004737 140132          BEQ     5$             ;YES GO ON
11547 055562 104003          CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11548                                ERROR   +3             ;FPP ERROR
11549 055564 077107          ;NO GO TO ERROR
11550 055566 020327 000204          SOB     R1,4$          ;ARE WE DONE
11551 055572 001403          CMP     R3,@204        ;IS FPS CORRECT
11552 055574 004737 140132          BEQ     6$             ;YES GO ON
11553 055600 104003          CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11554                                ERROR   +3             ;FPP ERROR
11555 055602 012637 000004          ;NO GO TO ERROR
11556 055606 000167 000010          MOV      (SP)+,@4      ;RESTORE VECTOR
11557                                JMP     FIN20
11558 055612 004737 140132          ;TSF20: CALL @DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
11559 055616 104003          ERROR   +3             ;FPP ERROR
11560                                ;ODD ADDRESS TRAP
11561 055620 000006          RTT                    ;RETURN
11562                                ;
11563 055622 000240          ;FIN20: NOP
11564                                ;
11565                                ;
11566                                ;
11567                                ;
11568                                ;
11569                                ;-----
11570                                ;TEST  FDST SOP MODE 7
11571                                ;
11572 055624          ;TSFP21:
11573                                ;
11574 055624 005037 140054          CLR      @TRPFLG          ;CLEAR TRAP FLAG
11575 055630 013746 000004          MOV      @4,-(SP)        ;SAVE TIMEOUT VECTOR

```


FLOATING POINT TESTS

```

11635
11636 056044 005037 140054      ;
11637 056050 013746 000004      CLR      @TRPFLG      ;CLEAR TRAP FLAG
11638 056054 012737 056216 000004      MOV      @4,-(SP)    ;SAVE TIME OUT VECTOR
11639 056062 012702 000200      MOV      @TSF22,@4   ;SETUP NEW VECTOR
11640 056066 170102      MOV      @200,R2     ;SETUP TO LOAD FPS
11641 056070 012705 000010      LDFPS   R2          ;SET FD=1
11642 056074 012704 003162      MOV      @8.,R5      ;INIT COUNTER
11643 056100 012724 177777      MOV      @TSTLOC,R4  ;SETUP POINTER TO TEST LOCATION
11644 056104 077503      MOV      @177777,(R4)+ ;INIT TEST LOCATION
11645 056106 012737 003172 003162      SOB      R5,100$    ;ARE WE DONE
11646 056114 170437 003162      MOV      @TSTLOC+10,@TSTLOC ;INIT TEST LOCATION
11647 056120 170203      CLRD    @TSTLOC     ; TEST INSTRUCTION
11648 056122 012702 003162      STFPS   R3          ;GET FPS
11649 056126 012701 000004      MOV      @TSTLOC,R2  ;SETUP POINTER TO DATA
11650 056132 022227 000000      MOV      @4,R1       ;INIT COUNTER
11651 056136 001403      1$:     CMP      (R2)+,@0  ;IS DATA CORRECT
11652 056140 004737 140132      BEQ      2$          ;YES GO ON
11653 056144 104003      CALL    @DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
11654      ERROR      +3      ;FPP ERROR
11655 056146 077107      2$:     SOB      R1,1$    ;NO GO TO ERROR
11656 056150 012701 000004      MOV      @4,R1       ;ARE WE DONE
11657 056154 022227 177777      3$:     CMP      (R2)+,@177777 ;INIT COUNTER
11658 056160 001403      BEQ      4$          ;IS DATA CORRECT
11659 056162 004737 140132      CALL    @DETFPA     ;YES GO ON
11660 056166 104003      ERROR      +3      ;DETERMINE FLOATING POINT FAULT. $$$
11661      ;FPP ERROR
11662 056170 077107      4$:     SOB      R1,3$    ;NO GO TO ERROR
11663 056172 020327 000204      CMP      R3,@204    ;ARE WE DONE
11664 056176 001403      BEQ      5$          ;CHECK FPS
11665 056200 004737 140132      CALL    @DETFPA     ;OK GO ON
11666 056204 104003      ERROR      +3      ;DETERMINE FLOATING POINT FAULT. $$$
11667      ;FPP ERROR
11668 056206 012637 000004      5$:     MOV      (SP)+,@4   ;NO GO TO ERROR
11669 056212 000167 000010      JMP      FIN2        ;RESTORE VECTOR
11670      ;
11671 056216 004737 140132      TSF22:  CALL    @DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
11672 056222 104003      ERROR      +3      ;FPP ERROR
11673      ;ODD ADDRESS TRAP
11674 056224 000006      RTT          ;RETURN
11675      ;
11676 056226 000240      FIN2:  NOP
11677      ;
11680      ;
11681      ;
11682      ;-----
11683      ;TEST  FDST SOP MODE 6 GR7
11684      ;
11685 056230      TSFP23:
11686      ;
11687 056230 005037 140054      CLR      @TRPFLG     ;CLEAR TRAP FLAG
11688 056234 013746 000004      MOV      @4,-(SP)    ;SAVE TIMEOUT VECTOR
11689 056240 012737 056352 000004      MOV      @TSF23,@4   ;SETUP NEW VECTOR
11690 056246 012702 000200      MOV      @200,R2     ;SETUP TO LOAD FPS
11691 056252 170102      LDFPS   R2          ;SET FD=1
11692 056254 012705 000004      MOV      @4,R5      ;INIT COUNTER
11693 056260 012704 003162      MOV      @TSTLOC,R4  ;SETUP POINTER TO TEST LOCATION

```

FLOATING POINT TESTS

```

11694 056264 012724 177777      100$:  MOV    #177777,(R4)      ;INIT TEST LOCATION
11695 056270 077503              SOB    R5,100$          ;ARE WE DONE
11696 056272 170467 124664      CLRD   TSTLOC           ; TEST INSTRUCTION
11697 056276 170203              STFPS  R3               ;GET FPS
11698 056300 012701 000004      MOV    #4,R1           ;INIT COUNTER
11699 056304 012702 003162      MOV    #TSTLOC,R2      ;SETUP POINTER TO DATA
11700 056310 022227 000000      1$:   CMP    (R2)+,#0     ;IS DATA CORRECT
11701 056314 001403              BEQ    2$              ;YES GO ON
11702 056316 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11703 056322 104003              ERROR  +3              ;FPP ERROR
11704                                ;NO GO TO ERROR
11705 056324 077107      2$:   SOB    R1,1$          ;ARE WE DONE
11706 056326 020327 000204      CMP    R3,#204         ;CHECK FPS
11707 056332 001403              BEQ    3$              ;OK GO ON
11708 056334 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11709 056340 104003              ERROR  +3              ;FPP ERROR
11710                                ;NO GO TO ERROR
11711 056342 012637 000004      3$:   MOV    (SP)+,#@4    ;RESTORE VECTOR
11712 056346 000167 000010      JMP    FIN23
11713                                ;
11714 056352 004737 140132      TSF23: CALL @DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
11715 056356 104003              ERROR  +3              ;FPP ERROR
11716                                ;ODD ADDRESS TRAP
11717 056360 000006              RTT
11718 056362 000240      FIN23: NOP
11719                                ;
11720                                ;
11721                                ;-----
11722                                ;TEST FDST SOP MODE 7 GR7
11723                                ;
11724                                ;
11725                                ;
11726                                ;
11727 056364      TSFP24:
11728                                ;
11729 056364 005037 140054      CLR    @TRPFLG         ;CLEAR TRAP FLAG
11730 056370 013746 000004      MOV    @4,-(SP)        ;SAVE TIME/VT VECTOR
11731 056374 012737 056552 000004  MOV    #TSF24,@@4      ;SETUP NEW VECTOR
11732 056402 012702 000200      MOV    #200,R2         ;SETUP TO LOAD FPS
11733 056406 170102              LDFPS  R2              ;SET FD=1
11734 056410 012705 000010      MOV    #8.,R5          ;INIT COUNTER
11735 056414 012704 003162      MOV    #TSTLOC,R4      ;SETUP TEST LOCATION POINTER
11736 056420 012724 177777      100$: MOV    #177777,(R4)  ;INIT TEST LOCATION
11737 056424 077503              SOB    R5,100$          ;ARE WE DONE
11738 056426 012737 003172 003172  MOV    #TSTLOC,@TSTLOC+10 ;INIT TEST LOCATION
11739 056434 170477 124532      CLRD   @TSTLOC+10     ; TEST INSTRUCTION
11740 056440 170203              STFPS  R3               ;GET FPS
11741 056442 012702 003162      MOV    #TSTLOC,R2      ;SETUP POINTER TO DATA
11742 056446 012701 000004      MOV    #4,R1           ;INIT COUNTER
11743 056452 022227 000000      1$:   CMP    (R2)+,#0     ;IS DATA CORRECT
11744 056456 001403              BEQ    2$              ;YES GO ON
11745 056460 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11746 056464 104003              ERROR  +3              ;FPP ERROR
11747                                ;NO GO TO ERROR
11748 056466 077107      2$:   SOB    R1,1$          ;ARE WE DONE
11749 056470 022227 003162      CMP    (R2)+,#TSTLOC   ;IS DATA CORRECT
11750 056474 001403              BEQ    3$              ;YES GO ON
11751 056476 004737 140132      CALL   @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11752 056502 104003              ERROR  +3              ;FPP ERROR

```

FLOATING POINT TESTS

```

11753
11754 056504 012701 000003      3$:  MOV    #3,R1          ;NO GO TO ERROR
11755 056510 022227 177777      4$:  CMP    (R2)+,#177777  ;INIT COUNTER
11756 056514 001403              BEQ    5$                ;IS DATA CORRECT
11757 056516 004737 140132      CALL   @@DETFPA         ;YES GO ON
11758 056522 104003              ERROR  +3               ;DETERMINE FLOATING POINT FAULT. $$$
11759
11760 056524 077107      5$:  SOB    R1,4$          ;FPP ERROR
11761 056526 020327 000204      CMP    R3,#204         ;NO GO TO ERROR
11762 056532 001403              BEQ    6$                ;ARE WE DONE
11763 056534 004737 140132      CALL   @@DETFPA         ;CHECK FPS
11764 056540 104003              ERROR  +3               ;OK GO ON
11765
11766 056542 012637 000004      6$:  MOV    (SP)+,#@4      ;DETERMINE FLOATING POINT FAULT. $$$
11767 056546 000167 000010      JMP    FIN24            ;FPP ERROR
11768
11769 056552 004737 140132      ;TSF24: CALL @@DETFPA   ;NO GO TO ERROR
11770 056556 104003              ERROR  +3               ;RESTORE VECTOR
11771
11772 056560 000006              RTT                    ;DETERMINE FLOATING POINT FAULT. $$$
11773
11774 056562 000240      ;FIN24: NOP            ;FPP ERROR
11775
11776
11777
11778
11779
11780
11781
11782
11783 056564      ;TSFP25:
11784
11785 056564 005037 140054      ;
11786 056570 005002              CLR    @@TRPFLG        ;CLEAR TRAP FLAG
11787 056572 170102              CLR    R2              ;SETUP TO LOAD FPS
11788 056574 012705 000004      LDFPS R2              ;SET FD=0
11789 056600 012704 003162      MOV    #4,R5          ;INIT COUNTER
11790 056604 012724 177777      100$: MOV    @TSTLOC,R4    ;SETUP POINTER TO TEST LOCATION
11791 056610 077503              SOB    #177777,(R4)+  ;INIT TEST LOCATION
11792 056612 012702 003162      MOV    R5,100$       ;ARE WE DONE
11793 056616 170422              MOV    @TSTLOC,R2     ;SETUP POINTER TO DATA
11794 056620 170203              CLRF  (R2)+          ; TEST INSTRUCTION
11795 056622 020227 003166      STFPS R3              ;GET FPS
11796 056626 001403              CMP    R2,@TSTLOC+4   ;IS R2 CORRECT
11797 056630 004737 140132      BEQ    1$              ;YES GO ON
11798 056634 104003              CALL   @@DETFPA         ;DETEPMINE FLOATING POINT FAULT. $$$
11799
11800 056636 012702 003162      1$:  MOV    @TSTLOC,R2     ;FPP ERROR
11801 056642 012701 000002      MOV    #2,R1          ;NO GO TO ERROR
11802 056646 022227 000000      2$:  CMP    (R2)+,#0     ;SETUP POINTER TO DATA
11803 056652 001403              BEQ    3$              ;INIT COUNTER
11804 056654 004737 140132      CALL   @@DETFPA         ;IS DATA CORRECT
11805 056660 104003              ERROR  +3               ;YES GO ON
11806
11807 056662 077107      3$:  SOB    R1,2$          ;DETERMINE FLOATING POINT FAULT. $$$
11808 056664 012701 000002      MOV    #2,R1          ;FPP ERROR
11809 056670 022227 177777      4$:  CMP    (R2)+,#177777 ;NO GO TO ERROR
11810 056674 001403              BEQ    5$                ;ARE WE DONE
11811 056676 004737 140132      CALL   @@DETFPA         ;INIT COUNTER

```

FLOATING POINT TESTS

```

11812 056702 104003          ERROR +3          ;FPP ERROR
11813                               ;NO GO TO ERROR
11814 056704 077107          5$: SOB R1,4$          ;ARE WE DONE
11815 056706 020327 000004  CMP R3,04          ;CHECK FPS
11816 056712 001403          BEQ 6$             ;OK GO ON
11817 056714 004737 140132  CALL @0DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11818 056720 104003          ERROR +3          ;FPP ERROR
11819                               ;NO GO TO ERROR
11820 056722          6$:
11821                               ;
11824                               ;
11825                               ;-----
11826                               ;TEST TSTF AND TSTD
11827                               ;
11828                               ;
11829 056722          TSFP26:
11830                               ;
11831 056722 005037 140054  CLR @0TRPFLG      ;CLEAR TRAP FLAG
11832 056726 005004          CLR R4              ;SETUP TO LOAD FPS
11833 056730 170104          LDFPS R4           ;SET FD=0
11834 056732 170567 000300  TSTF TS26D0      ; TEST INSTRUCTION
11835 056736 170203          STFPS R3           ;GET FPS
11836 056740 020327 000004  CMP R3,04          ;CHECK FPS
11837 056744 001403          BEQ 1$             ;OK GO ON
11838 056746 004737 140132  CALL @0DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11839 056752 104003          ERROR +3          ;FPP ERROR
11840                               ;NO GO TO ERROR
11841 056754 012704 057236  1$: MOV @TS26D0,R4    ;SETUP POINTERS TO DATA
11842 056760 012702 057266  MOV @TS26D3,R2    ;
11843 056764 004767 000224  JSR PC,CHEC26    ;CHECK IF DATA IS CORRECT
11844 056770 170537 057246  TSTF @0TS26D1    ; TEST INSTRUCTION
11845 056774 170203          STFPS R3           ;GET FPS
11846 056776 020327 000010  CMP R3,010       ;CHECK FPS
11847 057002 001403          BEQ 2$             ;OK GO ON
11848 057004 004737 140132  CALL @0DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11849 057010 104003          ERROR +3          ;FPP ERROR
11850                               ;NO GO TO ERROR
11851 057012 012704 057246  2$: MOV @TS26D1,R4    ;SETUP POINTERS TO DATA
11852 057016 012702 057276  MOV @TS26D4,R2    ;
11853 057022 004767 000166  JSR PC,CHEC26    ;CHECK IF DATA IS CORRECT
11854 057026 170567 000224  TSTF TS26D2      ; TEST INSTRUCTION
11855 057032 170203          STFPS R3           ;GET FPS
11856 057034 020327 000000  CMP R3,0         ;CHECK FPS
11857 057040 001403          BEQ 3$             ;OK GO ON
11858 057042 004737 140132  CALL @0DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
11859 057046 104003          ERROR +3          ;FPP ERROR
11860                               ;NO GO TO ERROR
11861 057050 012704 057256  3$: MOV @TS26D2,R4    ;SETUP POINTERS TO DATA
11862 057054 012702 057306  MOV @TS26D5,R2    ;
11863 057060 004767 000130  JSR PC,CHEC26    ;CHECK IF DATA IS CORRECT
11864 057064 012704 000200  MOV @200,R4       ;SETUP TO LOAD FPS
11865 057070 170104          LDFPS R4           ;SET FD=1
11866 057072 170537 057236  TSTD @0TS26D0    ; TEST INSTRUCTION
11867 057076 170203          STFPS R3           ;GET FPS
11868 057100 020327 000204  CMP R3,0204      ;CHECK FPS
11869 057104 001403          BEQ 4$             ;OK GO ON
11870 057106 004737 140132  CALL @0DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$

```

FLOATING POINT TESTS

```

11871 057112 104003          ERROR      +3          ;FPP ERROR
11872                                     ;NO GO TO ERROR
11873 057114 012704 057236 4$:  MOV      #TS26D0,R4          ;SETUP POINTERS TO DATA
11874 057120 012702 057266  MOV      #TS26D3,R2          ;
11875 057124 004767 000064  JSR      PC,CHEC26          ;CHECK IF DATA IS CORRECT
11876 057130 170567 000112  TSTD    TS26D1             ; TEST INSTRUCTION
11877 057134 170203          STFPS   R3                 ;GET FPS
11878 057136 020327 000210  CMP     R3,#210           ;CHECK FPS
11879 057142 001403          BEQ     5$                 ;OK GO ON
11880 057144 004737 140132  CALL   @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
11881 057150 104003          ERROR      +3          ;FPP ERROR
11882                                     ;NO GO TO ERROR
11883 057152 012704 057246 5$:  MOV      #TS26D1,R4          ;SETUP POINTERS TO DATA
11884 057156 012702 057276  MOV      #TS26D4,R2          ;
11885 057162 004767 000026  JSR      PC,CHEC26          ;CHECK IF DATA IS CORRECT
11886 057166 170567 000064  TSTD    TS26D2             ; TEST INSTRUCTION
11887 057172 170203          STFPS   R3                 ;GET FPS
11888 057174 020327 000200  CMP     R3,#200           ;CHECK FPS
11889 057200 001403          BEQ     6$                 ;OK GO ON
11890 057202 004737 140132  CALL   @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
11891 057206 104003          ERROR      +3          ;FPP ERROR
11892                                     ;NO GO TO ERROR
11893 057210          6$:  JMP      FIN26
11894 057210 000167 000102          ;
11895                                     ;
11896 057214 012701 000004  CHEC26: MOV     #4,R1          ;INIT COUNTER
11897 057220 022422          1$:  CMP     (R4)+,(R2)+      ;IS DATA CORRECT
11898 057222 001403          BEQ     2$                 ;YES GO ON
11899 057224 004737 140132  CALL   @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
11900 057230 104003          ERROR      +3          ;FPP ERROR
11901                                     ;NO GO TO ERROR
11902 057232 077106          2$:  SOB     R1,1$
11903 057234 000207          RTS     PC                 ;ARE WE DONE
11904                                     ;RETURN
11905 057236 000177          ;
11906 057240 177777          TS2600: .WORD 177
11907 057242 177777          .WORD 177777
11908 057244 177777          .WORD 177777
11909 057246 177777          .WORD 177777
11910 057250 000000          TS26D1: .WORD 177777
11911 057252 000000          .WORD 0
11912 057254 000000          .WORD 0
11913 057256 077777          TS26D2: .WORD 77777
11914 057260 000000          .WORD 0
11915 057262 000000          .WORD 0
11916 057264 000000          .WORD 0
11917 057266 000177          TS26D3: .WORD 177
11918 057270 177777          .WORD 177777
11919 057272 177777          .WORD 177777
11920 057274 177777          .WORD 177777
11921 057276 177777          TS26D4: .WORD 177777
11922 057300 000000          .WORD 0
11923 057302 000000          .WORD 0
11924 057304 000000          .WORD 0
11925 057306 077777          TS26D5: .WORD 77777
11926 057310 000000          .WORD 0
11927 057312 000000          .WORD 0
    
```

FLOATING POINT TESTS

```

11928 057314 000000          .WORD 0
11929 057316 000240          FIN26: NOP
11930                          ;
11931                          ;
11932                          ;-----
11933                          ;TEST ABSF
11934                          ;
11935                          ;
11936                          ;
11937                          ;
11938 057320          TSFP27:
11939                          ;
11940 057320 005037 140054          CLR      @TRPFLG          ;CLEAR TRAP FLAG
11941 057324 005005          CLR      R5              ;SETUP TO LOAD FPS
11942 057326 170105          LDFPS   R5              ;SET FD=0
11943 057330 012701 000014          MOV      @12.,R1         ;INIT COUNTER
11944 057334 012704 003162          MOV      @TSTLOC,R4     ;SETUP POINTER TO TEST LOCATION
11945 057340 012703 057600          MOV      @TS27D0,R3     ;SETUP POINTER TO TEST VALUE
11946 057344 012324          100$: MOV      (R3)+,(R4)+    ;INIT TEST LOCATION
11947 057346 077102          SOB      R1,100$       ;ARE WE DONE
11948 057350 012705 003162          MOV      @TSTLOC,R5     ;SETUP POINTER TO DATA
11949 057354 170615          ABSF    (R5)           ; TEST INSTRUCTION
11950 057356 170203          STFPS   R3              ;GET FPS
11951 057360 020527 003162          CMP      R5,@TS1LOC     ;IS R5 CORRECT
11952 057364 001403          BEQ     1$              ;YES GO ON
11953 057366 004737 140132          CALL    @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11954 057372 104003          ERROR   +3             ;FPP ERROR
11955                          ;NO GO TO ERROR
11956 057374 012702 057630          1$:  MOV      @TS27D3,R2   ;SETUP POINTER TO DATA
11957 057400 004767 000152          JSR     PC,CHEC27       ;CHECK IF DATA IS CORRECT
11958 057404 020327 000000          CMP     R3,@0           ;CHECK FPS
11959 057410 001403          BEQ     2$              ;OK GO ON
11960 057412 004737 140132          CALL    @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11961 057416 104003          ERROR   +3             ;FPP ERROR
11962                          ;NO GO TO ERROR
11963 057420 012705 003172          2$:  MOV      @TSTLOC+10,R5 ;SETUP POINTER TO DATA
11964 057424 170625          ABSF    (R5)+          ; TEST INSTRUCTION
11965 057426 170203          STFPS   R3              ;GET FPS
11966 057430 020527 003176          CMP     R5,@TSTLOC+14  ;IS R5 CORRECT
11967 057434 001403          BEQ     3$              ;YES GO ON
11968 057436 004737 140132          CALL    @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11969 057442 104003          ERROR   +3             ;FPP ERROR
11970                          ;NO GO TO ERROR
11971 057444 012705 003172          3$:  MOV      @TSTLOC+10,R5 ;SETUP POINTER TO DATA
11972 057450 012702 057640          MOV      @TS27D4,R2     ;
11973 057454 004767 000076          JSR     PC,CHEC27       ;CHECK IF DATA IS CORRECT
11974 057460 020327 000000          CMP     R3,@0           ;CHECK FPS
11975 057464 001403          BEQ     4$              ;OK GO ON
11976 057466 004737 140132          CALL    @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11977 057472 104003          ERROR   +3             ;FPP ERROR
11978                          ;NO GO TO ERROR
11979 057474 012705 003162          4$:  MOV      @TSTLOC,R5    ;SETUP POINTER TO DATA
11980 057500 170665 000020          ABSF    20(R5)         ; TEST INSTRUCTION
11981 057504 170203          STFPS   R3              ;GET FPS
11982 057506 020527 003162          CMP     R5,@TSTLOC     ;IS R5 CORRECT
11983 057512 001403          BEQ     5$              ;YES GO ON
11984 057514 004737 140132          CALL    @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11985 057520 104003          ERROR   +3             ;FPP ERROR
11986                          ;NO GO TO ERROR

```


FLOATING POINT TESTS

```

11987 057522 012705 003202      5$:   10V   #TSTLOC+20,R5           ;SETUP POINTERS TO DATA
11988 057526 012702 057650      MOV   #TS27D5,R2           ;
11989 057532 004767 000020      JSR   PC,CHEC27           ;CHECK IF DATA IS CORRECT
11990 057536 020327 000004      CMP   R3,#4              ;CHECK FPS
11991 057542 001403              BEQ   6$                 ;OK GO ON
11992 057544 004737 140132      CALL  @#DEFPA           ;DETERMINE FLOATING POINT FAULT. $$$
11993 057550 104003              ERROR +3                ;FPP ERROR
11994                               ;NO GO TO ERROR
11995 057552              6$:   JMP   FIN27
11996 057552 000167 000102              ;
11997                               ;
11998 057556 012701 000004      CHEC27: MOV #4,R1          ;INIT COUNTER
11999 057562 022522              1$:   CMP   (R5)+,(R2)+    ;IS DATA CORRECT
12000 057564 001403              BEQ   2$                 ;YES GO ON
12001 057566 004737 140132      CALL  @#DEFPA           ;DETERMINE FLOATING POINT FAULT. $$$
12002 057572 104003              ERROR +3                ;FPP ERROR
12003                               ;NO GO TO ERROR
12004 057574 077106              2$:   SOB  R1,1$         ;ARE WE DONE
12005 057576 000207              RTS   PC                 ;RETURN
12006                               ;
12007 057600 177777      TS27D0: .WORD 177777
12008 057602 177777      .WORD 177777
12009 057604 177777      .WORD 177777
12010 057606 177777      .WORD 177777
12011 057610 000377      TS27D1: .WORD 377
12012 057612 175436      .WORD 175436
12013 057614 136477      .WORD 136477
12014 057616 000001      .WORD 1
12015 057620 000177      TS27D2: .WORD 177
12016 057622 175436      .WORD 175436
12017 057624 136477      .WORD 136477
12018 057626 000001      .WORD 1
12019 057630 077777      TS27D3: .WORD 77777
12020 057632 177777      .WORD 177777
12021 057634 177777      .WORD 177777
12022 057636 177777      .WORD 177777
12023 057640 000377      TS27D4: .WORD 377
12024 057642 175436      .WORD 175436
12025 057644 136477      .WORD 136477
12026 057646 000001      .WORD 1
12027 057650 000000      TS27D5: .WORD 0
12028 057652 000000      .WORD 0
12029 057654 136477      .WORD 136477
12030 057656 000001      .WORD 1
12031 057660 000240      FIN27: NOP
12032                               ;
12033                               ;
12034                               ;
12035                               ;
12036                               ;
12037                               ;TEST ABSD
12038                               ;
12039                               ;
12040 057662      TSFP30:
12041                               ;
12042 057662 005037 140054      CLR   @#TRPFLG          ;CLEAR TRAP FLAG
12043 057666 012705 000200      MOV   #200,R5           ;SETUP TO LOAD FPS
12044 057672 170105              LDFPS R5                ;SET FD=1
12045 057674 012701 000014      MOV   #12.,R1          ;INIT COUNTER

```

FLOATING POINT TESTS

```

12046 057700 012704 003162      MOV      #TSTLOC,R4      ;SETUP POINTER TO TEST LOCATION
12047 057704 012703 060144      MOV      #TS3000,R3     ;SETUP POINTER TO TEST VALUE
12048 057710 012324      100$:   MOV      (R3)+,(R4)+   ;INIT TEST LOCATION
12049 057712 077102      SOB      R1,100$       ;ARE WE DONE
12050 057714 012705 003162      MCV      #TSTLOC,R5     ;SETUP POINTER TO DATA
12051 057720 170615      ABSD    (R5)           ; TEST INSTRUCTION
12052 057722 170203      STFPS   R3             ;GET FPS
12053 057724 020527 003162      CMP      R5,#TSTLOC    ;IS R5 CORRECT
12054 057730 001403      BEQ     1$             ;YES GO ON
12055 057732 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12056 057736 104003      ERROR   +3            ;FPP ERROR
12057                                ;NO GO TO ERROR
12058 057740 012702 060174      1$:     MOV      #TS3003,R2     ;SETUP POINTER TO DATA
12059 057744 004767 000152      JSR     PC,CHEC30      ;CHECK IF DATA IS CORRECT
12060 057750 020327 000200      CMP     R3,#200        ;CHECK FPS
12061 057754 001403      BEQ     2$             ;OK GO ON
12062 057756 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12063 057762 104003      ERROR   +3            ;FPP ERROR
12064                                ;NO GO TO ERROR
12065 057764 012705 003172      2$:     MOV      #TSTLOC+10,R5  ;SETUP POINTER TO DATA
12066 057770 170625      ABSD    (R5)+         ; TEST INSTRUCTION
12067 057772 170203      STFPS   R3             ;GET FPS
12068 057774 020527 003202      CMP     R5,#TSTLOC+20 ;IS R5 CORRECT
12069 060000 001403      BEQ     3$             ;YES GO ON
12070 060002 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12071 060006 104003      ERROR   +3            ;FPP ERROR
12072                                ;NO GO TO ERROR
12073 060010 012705 003172      3$:     MOV      #TSTLOC+10,R5  ;SETUP POINTERS TO DATA
12074 060014 012702 060204      MOV      #TS3004,R2    ;
12075 060020 004767 000076      JSR     PC,CHEC30      ;CHECK IF DATA IS CORRECT
12076 060024 020327 000200      CMP     R3,#200        ;CHECK FPS
12077 060030 001403      BEQ     4$             ;OK GO ON
12078 060032 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12079 060036 104003      ERROR   +3            ;FPP ERROR
12080                                ;NO GO TO ERROR
12081 060040 012705 003162      4$:     MOV      #TSTLOC,R5     ;SETUP POINTER TO DATA
12082 060044 170665 000020      ABSD    20(R5)        ; TEST INSTRUCTION
12083 060050 170203      STFPS   R3             ;GET FPS
12084 060052 020527 003162      CMP     R5,#TSTLOC    ;IS R5 CORRECT
12085 060056 001403      BEQ     5$             ;YES GO ON
12086 060060 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12087 060064 104003      ERROR   +3            ;FPP ERROR
12088                                ;NO GO TO ERROR
12089 060066 012705 003202      5$:     MOV      #TSTLOC+20,R5  ;SETUP POINTERS TO DATA
12090 060072 012702 060214      MOV      #TS3005,R2    ;
12091 060076 004767 000020      JSR     PC,CHEC30      ;CHECK IF DATA IS CORRECT
12092 060102 020327 000204      CMP     R3,#204        ;CHECK FPS
12093 060106 001403      BEQ     6$             ;OK GO ON
12094 060110 004737 140132      CALL    @#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12095 060114 104003      ERROR   +3            ;FPP ERROR
12096                                ;NO GO TO ERROR
12097 060116      6$:     JMP      FIN30
12098 060116 000167 000102      ;
12099                                ;
12100 060122 012701 000004      CHEC30: MOV      #4,R1    ;INIT COUNTER
12101 060126 022522      1$:     CMP      (R5)+,(R2)+   ;IS DATA CORRECT
12102 060130 001403      BEQ     2$             ;YES GO ON

```

FLOATING POINT TESTS

```

12103 060132 004737 140132      CALL  @#DETFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12104 060136 104003              ERROR  +3          ; FPP ERROR
12105                               ; NO GO TO ERROR
12106 060140 077106      2$: SOB  R1,1$      ; ARE WE DONE
12107 060142 000207      RTS    PC          ; RETURN
12108                               ;
12109 060144 177777      TS30D0: .WORD 177777
12110 060146 177777      .WORD 177777
12111 060150 177777      .WORD 177777
12112 060152 177777      .WORD 177777
12113 060154 000377      TS30D1: .WORD 377
12114 060156 175436      .WORD 175436
12115 060160 136477      .WORD 136477
12116 060162 000001      .WORD 1
12117 060164 000177      TS30D2: .WORD 177
12118 060166 175436      .WORD 175436
12119 060170 136477      .WORD 136477
12120 060172 000001      .WORD 1
12121 060174 077777      TS30D3: .WORD 77777
12122 060176 177777      .WORD 177777
12123 060200 177777      .WORD 177777
12124 060202 177777      .WORD 177777
12125 060204 000377      TS30D4: .WORD 377
12126 060206 175436      .WORD 175436
12127 060210 136477      .WORD 136477
12128 060212 000001      .WORD 1
12129 060214 000000      TS30D5: .WORD 0
12130 060216 000000      .WORD 0
12131 060220 000000      .WORD 0
12132 060222 000000      .WORD 0
12133 060224 000240      FIN30: NOP
12134                               ;
12137                               ;
12138                               ; -----
12139                               ; TEST  FDST SOP MODE 2 GR7
12140                               ;
12141 060226      TSFP31:
12142                               ;
12143 060226 005037 140054      CLR  @#TRPFLG      ; CLEAR TRAP FLAG
12144 060232 013746 000004      MOV  @#4, -(SP)    ; SAVE TIMEOUT VECTOR
12145 060236 012737 060354 000004      MOV  @#TSF31, @#4 ; SETUP NEW VECTOR
12146 060244 012702 000200      MOV  @#200, R2     ; SETUP TO LOAD FPS
12147 060250 170102              LDFPS R2           ; SET FD=1
12148 060252 170527 000005      TSD31: TST  @#5    ; TEST INSTRUCTION
12149 060256 000240              NOP
12150 060260 000240              NOP
12151 060262 000240              NOP
12152 060264 170203              STFPS R3           ; GET FPS
12153 060266 020327 000204      CMP  R3, @#204    ; CHECK FPS
12154 060272 001403              BEQ  1$           ; OK GO ON
12155 060274 004737 140132      CALL  @#DETFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12156 060300 104003              ERROR  +3          ; FPP ERROR
12157                               ; NO GO TO ERROR
12158 060302 012702 060254      1$: MOV  @#TSD31+2, R2 ; SETUP POINTER TO DATA
12159 060306 022227 000005      CMP  (R2)+, @#5   ; IS DATA CORRECT
12160 060312 001403              BEQ  2$           ; YES GO ON
12161 060314 004737 140132      CALL  @#DETFPA      ; DETERMINE FLOATING POINT FAULT. $$$
    
```

FLOATING POINT TESTS

```

12162 060320 104003          ERROR  +3          ;FPP ERROR
12163                                ;NO GO TO ERROR
12164 060322 012701 000003    2$:  MOV  #3,R1          ;INIT COUNTER
12165 060326 022227 000240    3$:  CMP  (R2)+,#240     ;IS DATA CORRECT
12166 060332 001403          BEQ   4$          ;YES GO ON
12167 060334 004737 140132    CALL  @#DEFPA     ;DETERMINE FLOATING POINT FAULT. $$$
12168 060340 104003          ERROR  +3          ;FPP ERROR
12169                                ;NO GO TO ERROR
12170 060342 077107          4$:  SOB  R1,3$      ;ARE WE DONE
12171 060344 012637 000004    MOV  (SP)+,#4     ;RESTORE VECTOR
12172 060350 000167 000010    JMP  FIN31
12173                                ;
12174 060354 004737 140132    ;TSF31: CALL @#DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
12175 060360 104003          ERROR  +3          ;FPP ERROR
12176                                ;ODD ADDRESS TRAP
12177 060362 000006          RTT          ;RETURN
12178                                ;
12179 060364 000240          ;FIN31: NOP
12180                                ;
12181                                ;
12182                                ;-----
12183                                ;
12184                                ;
12185                                ;TEST NEGF
12186                                ;
12187                                ;
12188 060366          ;TSFP32:
12189                                ;
12190 060366 005037 140054    CLR  @#TRPFLG     ;CLEAR TRAP FLAG
12191 060372 005005          CLR  R5           ;SETUP TO LOAD FPS
12192 060374 170105          LDFPS R5         ;SET FD=0
12193 060376 012701 000014    MOV  #12.,R1     ;INIT COUNTER
12194 060402 012704 003162    MOV  #TSTLOC,R4  ;SETUP POINTER TO TEST LOCATION
12195 060406 012703 060576    MOV  #TS32D0,R3  ;SETUP POINTER TO TEST VALUE
12196 060412 012324          100$: MOV  (R3)+,(R4)+ ;INIT TEST LOCATION
12197 060414 077102          SOB  R1,100$     ;ARE WE DONE
12198 060416 170767 122540    NEGF TSTLOC      ; TEST INSTRUCTION
12199 060422 170203          STFPS R3        ;GET FPS
12200 060424 012705 003162    MOV  #TSTLOC,R5  ;SETUP POINTERS TO DATA
12201 060430 012702 060626    MOV  #TS32D3,R2  ;
12202 060434 004767 000114    JSR  PC,CHEC32   ;CHECK IF DATA IS CORRECT
12203 060440 020327 000000    CMP  R3,#0       ;CHECK FPS
12204 060444 001403          BEQ  1$          ;YES GO ON
12205 060446 004737 140132    CALL @#DEFPA     ;DETERMINE FLOATING POINT FAULT. $$$
12206 060452 104003          ERROR  +3          ;FPP ERROR
12207                                ;NO GO TO ERROR
12208 060454 170767 122512    1$:  NEGF TSTLOC+10 ; TEST INSTRUCTION
12209 060460 170203          STFPS R3        ;GET FPS
12210 060462 012705 003172    MOV  #TSTLOC+10,R5 ;SETUP POINTERS TO DATA
12211 060466 012702 060636    MOV  #TS32D4,R2  ;
12212 060472 004767 000056    JSR  PC,CHEC32   ;CHECK IF DATA IS CORRECT
12213 060476 020327 000010    CMP  R3,#10     ;CHECK FPS
12214 060502 001403          BEQ  2$          ;OK GO ON
12215 060504 004737 140132    CALL @#DEFPA     ;DETERMINE FLOATING POINT FAULT. $$$
12216 060510 104003          ERROR  +3          ;FPP ERROR
12217                                ;NO GO TO ERROR
12218 060512 170767 122464    2$:  NEGF TSTLOC+20 ; TEST INSTRUCTION
12219 060516 170203          STFPS R3        ;GET FPS
12220 060520 012705 003202    MOV  #TSTLOC+20,R5 ;SETUP POINTERS TO DATA

```

FLOATING POINT TESTS

```

12221 060524 012702 060646      MOV      #TS32D5,R2      ;
12222 060530 004767 000020      JSR      PC,CHEC32      ;CHECK TF DATA IS CORRECT
12223 060534 020327 000004      CMP      R3,#4          ;CHECK FPS
12224 060540 001403              BEQ      3$             ;OK GO ON
12225 060542 004737 140132      CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12226 060546 104003              ERROR    +3            ;FPP ERROR
12227                                ;NO GO TO ERROR
12228 060550              3$:
12229 060550 000167 000102              JMP      FIN32
12230                                ;
12231 060554 012701 000004      CHEC32: MOV    #4,R1      ;INIT COUNTER
12232 060560 022522              1$:  CMP    (R5)+,(R2)+  ;IS DATA CORRECT
12233 060562 001403              BEQ      2$             ;YES GO ON
12234 060564 004737 140132      CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12235 060570 104003              ERROR    +3            ;FPP ERROR
12236                                ;NO GO TO ERROR
12237 060572 077106              2$:  SOB    R1,1$        ;ARE WE DONE
12238 060574 000207              RTS      PC             ;RETURN
12239                                ;
12240 060576 170000      TS32D0: .WORD 170000
12241 060600 003541              .WORD 3541
12242 060602 177777              .WORD 177777
12243 060604 172710              .WORD 172710
12244 060606 070000      TS32D1: .WORD 70000
12245 060610 003541              .WORD 3541
12246 060612 177777              .WORD 177777
12247 060614 172710              .WORD 172710
12248 060616 000177              .WORD 177
12249 060620 100000              .WORD 100000
12250 060622 177777              .WORD 177777
12251 060624 177007              .WORD 177007
12252 060626 070000      TS32D3: .WORD 70000
12253 060630 003541              .WORD 3541
12254 060632 177777              .WORD 177777
12255 060634 172710              .WORD 172710
12256 060636 170000      TS32D4: .WORD 170000
12257 060640 003541              .WORD 3541
12258 060642 177777              .WORD 177777
12259 060644 172710              .WORD 172710
12260 060646 000000      TS32D5: .WORD 0
12261 060650 000000              .WORD 0
12262 060652 177777              .WORD 177777
12263 060654 177007              .WORD 177007
12264 060656 000240      FIN32:  NOP
12265                                ;
12266                                ;
12269                                ;
12270                                ;-----
12271                                ;TEST  NEG0
12272 060660      TSFP33:
12273                                ;
12274 060660 005037 140054      CLR      @#TRPFLG      ;CLEAR TRAP FLAG
12275 060664 012705 000200      MOV      #200,R5       ;SETUP TO LOAD FPS
12276 060670 170105              LDFPS   R5             ;SET FD=1
12277 060672 012701 000014      MOV      #12.,R1       ;INIT COUNTER
12278 060676 012704 003162      MOV      #TSTLOC,R4    ;SETUP POINTER TO TEST LOCATION
12279 060702 012703 061072      MOV      #TS33D0,R3    ;SETUP POINTER TO TEST VALUE

```

FLOATING POINT TESTS

```

12280 060706 012324      100$:  MOV      (R3)+,(R4)+      ;INIT TEST LOCATION
12281 060710 077102      SOB      R1,100$          ;ARE WE DONE
12282 060712 170767 122244  NEG0    TSTLOC           ; TEST INSTRUCTION
12283 060716 170203      STFPS   R3               ;GET FPS
12284 060720 012705 003162  MOV     #TSTLOC,R5       ;SETUP POINTERS TO DATA
12285 060724 012702 061122  MOV     #TS33D3,R2
12286 060730 004767 000114  JSR     PC,CHEC33        ;
12287 060734 020327 000200  CMP     R3,#200          ;CHECK IF DATA IS CORRECT
12288 060740 001403      BEQ     1$               ;CHECK FPS
12289 060742 004737 140132  CALL   @#DETFPA         ;OK GO ON
12290 060746 104003      ERROR   +3              ;DETERMINE FLOATING POINT FAULT. $$$
12291                                     ;FPP ERROR
12292 060750 170767 122216  1$:    NEG0    TSTLOC+10    ; TEST INSTRUCTION
12293 060754 170203      STFPS   R3               ;GET FPS
12294 060756 012705 003172  MOV     #TSTLOC+10,R5   ;SETUP POINTERS TO DATA
12295 060762 012702 061132  MOV     #TS33D4,R2
12296 060766 004767 000056  JSR     PC,CHEC33        ;
12297 060772 020327 000210  CMP     R3,#210          ;CHECK IF DATA IS CORRECT
12298 060776 001403      BEQ     2$               ;CHECK FPS
12299 061000 004737 140132  CALL   @#DETFPA         ;OK GO ON
12300 061004 104003      ERROR   +3              ;DETERMINE FLOATING POINT FAULT. $$$
12301                                     ;FPP ERROR
12302 061006 170767 122170  2$:    NEG0    TSTLOC+20    ; TEST INSTRUCTION
12303 061012 170203      STFPS   R3               ;GET FPS
12304 061014 012705 003202  MOV     #TSTLOC+20,R5   ;SETUP POINTERS TO DATA
12305 061020 012702 061142  MOV     #TS33D5,R2
12306 061024 004767 000020  JSR     PC,CHEC33        ;
12307 061030 020327 000204  CMP     R3,#204          ;CHECK IF DATA IS CORRECT
12308 061034 001403      BEQ     3$               ;CHECK FPS
12309 061036 004737 140132  CALL   @#DETFPA         ;OK GO ON
12310 061042 104003      ERROR   +3              ;DETERMINE FLOATING POINT FAULT. $$$
12311                                     ;FPP ERROR
12312 061044                                     ;NO GO TO ERROR
12313 061044 000167 000102  3$:    JMP     FIN33
12314                                     ;
12315 061050 012701 000004  CHEC33: MOV     #4,R1      ;INIT COUNTER
12316 061054 022522  1$:    CMP     (R5)+,(R2)+    ;IS DATA CORRECT
12317 061056 001403      BEQ     2$               ;YES GO ON
12318 061060 004737 140132  CALL   @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
12319 061064 104003      ERROR   +3              ;FPP ERROR
12320                                     ;NO GO TO ERROR
12321 061066 077106  2$:    SOB      R1,1$          ;ARE WE DONE
12322 061070 000207      RTS     PC               ;RETURN
12323                                     ;
12324 061072 170000  TS33D0: .WORD   170000
12325 061074 003541      .WORD   3541
12326 061076 177777      .WORD   177777
12327 061100 172710      .WORD   172710
12328 061102 070000  TS33D1: .WORD   70000
12329 061104 003541      .WORD   3541
12330 061106 177777      .WORD   177777
12331 061110 172710      .WORD   172710
12332 061112 000177  TS33D2: .WORD   177
12333 061114 100000      .WORD   100000
12334 061116 177777      .WORD   177777
12335 061120 177007      .WORD   177007
12336 061122 070000  TS33D3: .WORD   70000
    
```

FLOATING POINT TESTS

```

12337 061124 003541          .WORD 3541
12338 061126 177777          .WORD 177777
12339 061130 172710          .WORD 172710
12340 061132 170000          TS33D4: .WORD 170000
12341 061134 003541          .WORD 3541
12342 061136 177777          .WORD 177777
12343 061140 172710          .WORD 172710
12344 061142 000000          TS33D5: .WORD 0
12345 061144 000000          .WORD 0
12346 061146 000000          .WORD 0
12347 061150 000000          .WORD 0
12348 061152 000240          FIN33: NOP
12349                          ;
12352                          ;
12353                          ;
12354                          ;-----;
12355                          ;TEST LDD MODE 0, ILLEGAL AC7
12356                          ;
12357 061154          MFSRCMO: ;
12358                          ;
12359                          ;
12360 061154 012704 047600          MOV      #47600,R4          ;SETUP FPP STATUS
12361 061160 170104          LDFPS   R4                ;LOAD FP STATUS
12362 061162 012702 003122          MOV      #RECFEC,R2       ;POINT TO RECEIVED FEC MEMORY
12363 061166 172407          1$:    LDD      R7,AC0      ;*TEST INSTRUCTION
12364                          ;LOAD ACO FROM ILLEGAL AC7
12365 061170 170201          STFPS   R1                ;SAVE FPP STATUS
12366 061172 022701 147600          CMP      #147600,R1       ;VERIFY FER BIT SET
12367 061176 001403          BEQ     2$                ;BRANCH IF GOOD ERROR CONDITION
12368 061200 004737 140132          CALL    @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
12369 061204 104003          ERROR  +3                ;FPP ERROR
12370                          ;THE FER BIT DIDNT SET
12371 061206 170312          2$:    STST   (R2)         ;SAVE FEC AND FEA
12372 061210 022722 000002          CMP      #2,(R2)+        ;VERIFY FEC CONTENTS
12373 061214 001403          BEQ     3$                ;BRANCH IF GOOD
12374 061216 004737 140132          CALL    @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
12375 061222 104003          ERROR  +3                ;FPP ERROR
12376                          ;FEC NE 2 (OPCODE ERROR)
12377 061224 022722 061166          3$:    CMP      #1$,(R2)+  ;VERFIY FEA CONTENTS
12378 061230 001403          BEQ     4$                ;BRANCH FI GOOD
12379 061232 004737 140132          CALL    @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
12380 061236 104003          ERROR  +3                ;FPP ERROR
12381                          ;FEA NOT CORRECT ERROR ADDRESS
12382 061240          4$:
12385                          ;
12386                          ;-----;
12387                          ;TEST LDD MODE2
12388                          ;
12389 061240          MLDDM2: ;
12390                          ;
12391 061240 012701 003142          MOV      #RECDST,R1       ;POINT TO RECEIVED DATA LOCATION
12392 061244 012704 003234          MOV      #TAB1,R4        ;POINT TO GOOD DATA
12393 061250 012702 047750          MOV      #47750,R2       ;LOAD GOOD STATUS
12394 061254 170102          LDFPS   R2                ;LOAD FPP STATUS - DOUBLE.ID
12395 061256 172424          LDD     (R4)+,AC0        ;*TEST INSTRUCTION - MODE 2
12396 061260 170203          STFPS   R3                ;SAVE TEST FPP STATUS
12397 061262 174021          STD     ACO,(R1)+        ;SAVE TEST RESULT MODE 2

```

FLOATING POINT TESTS

```

12398 061264 020203          CMP      R2,R3          ;VERIFY FPP STATUS
12399 061266 001403          BEQ      1$            ;BRANCH IF GOOD
12400 061270 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT $$$
12401 061274 104003          ERROR    +3            ;FPP ERROR
12402                                ;BAD FPP STSUS
12403 061276 022704 003244    1$:    CMP      @TAB1+10,R4 ;VERFIY AUTO-INCR
12404 061302 001403          BEQ      2$            ;BRANCH IF GOOD
12405 061304 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12406 061310 104003          ERROR    +3            ;FPP ERROR
12407                                ;BAD AUTO-INCR
12408 061312 012704 003234    2$:    MOV      @TAB1,R4      ;POINT TO RECEIVED DATA
12409 061316 162701 000010    SUB      @10,R1        ;RETURN R1 TO PROPER VALUE
12410 061322 004767 056552    JSR     R7,DATVER      ;VERFIY DATA FROM FPP
12411 061326 005767 121566    TST     COUNT          ;SEE IF COUNTER=0
12412 061332 001403          BEQ      3$            ;BRANCH IF GOOD COMPARE
12413 061334 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12414 061340 104003          ERROR    +3            ;FPP ERROR
12415 061342                                ;BAD DATA FROM FPP
12416                                ;
12419                                ;
12420                                ;-----
12421                                ;TEST LDD MODE 3
12422                                ;
12423 061342                                ;MLDDM3:
12424                                ;
12425 061342 012737 003142 003164    MOV      @RECDST,@TSTLOC+2 ;POINT TO RECEIVED DATA LOCATION
12426 061350 012701 003164          MOV      @TSTLOC+2,R1   ;SETUP STD IN MODE 3
12427 061354 012737 003244 003162    MOV      @TAB2,@TSTLOC  ;POINT TO DATA TABLE
12428 061362 012704 003162          MOV      @TSTLOC,R4    ;POINT TO GOOD DATA
12429 061366 012702 047750          MOV      @47750,R2     ;LOAD GOOD STATUS
12430 061372 170102          LDFPS   R2             ;LOAD FPP STATUS - DOUBLE.ID
12431 061374 172434          LDD     @R4+ ,ACO     ;*TEST INSTRUCTION - MODE 2
12432 061376 170203          STFPS   R3             ;SAVE TEST FPP STATUS
12433 061400 174031          STD     ACO,@R1+      ;SAVE TEST RESULT IN MODE 3
12434 061402 022703 047740          CMP      @47740,R3     ;VERIFY FPP STATUS
12435 061406 001403          BEQ      1$            ;BRANCH IF GOOD
12436 061410 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12437 061414 104003          ERROR    +3            ;FPP ERROR
12438                                ;BAD FPP STSUS
12439 061416 022704 003164    1$:    CMP      @TSTLOC+2,R4  ;VERFIY AUTO-INCR
12440 061422 001403          BEQ      2$            ;BAD AUTO-DEC ON LDD
12441 061424 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12442 061430 104003          ERROR    +3            ;FPP ERROR
12443                                ;BAD AUTO-INC
12444 061432 022701 003166    2$:    CMP      @TSTLOC+4,R1  ;TEST STD AUTO-INC
12445 061436 001403          BEQ      3$            ;BRANCH IF GOOD
12446 061440 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12447 061444 104003          ERROR    +3            ;FPP ERROR
12448                                ;BAD AUTO-INCR
12449 061446 012704 003244    3$:    MOV      @TAB2,R4      ;POINT TO RECEIVED DATA
12450 061452 012701 003142    MOV      @RECDST,R1    ;POINT TO RECEIVED DATA
12451 061456 004767 056416    JSR     R7,DATVER      ;VERFIY DATA FROM FPP
12452 061462 005767 121432    TST     COUNT          ;SEE IF COUNTER=0
12453 061466 001403          BEQ      4$            ;BRANCH IF GOOD COMPARE
12454 061470 004737 140132    CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12455 061474 104003          ERROR    +3            ;FPP ERROR
12456                                ;BAD DATA FROM FPP

```


FLOATING POINT TESTS

```

12457 061476      4$:
12458             ;
12461             ;
12462             ;
12463             ;-----
12464             ;TEST LDF, STD MODE 4
12465 061476      ;
12466             ;MLDDM4:
12467 061476 012701 003146      ;      MOV      #RECDST+4,R1      ;POINT TO RECEIVED DATA LOCATION
12468 061502 012704 003260      ;      MOV      #TAB3+4,R4      ;POINT TO GOOD DATA
12469 061506 012705 003314      ;      MOV      #TAB6,R5      ;CLEAR OUT ACO
12470 061512 170127 000200      ;      LDFPS   #200      ;SET TO DOUBLE
12471 061516 172415             ;      LDD      (R5),ACO      ;ACO=0
12472 061520 012702 047550      ;      MOV      #47550,R2      ;LOAD GOOD STATUS FLOATING
12473 061524 170102             ;      LDFPS   R2      ;LOAD FPP STATUS - DOUBLE, ID
12474 061526 172444             ;      LDF      -(R4),ACO      ;*TEST INSTRUCTION - MODE 4
12475 061530 170203             ;      STFPS   R3      ;SAVE TEST FPP STATUS
12476 061532 012702 047750      ;      MOV      #47750,R2      ;SET TO DOUBLE MODE
12477 061536 170102             ;      LDFPS   R2      ;SET FPP TO DOUBLE
12478 061540 174041             ;      STD     ACO, -(R1)      ;SAVE TEST RESULT
12479 061542 022703 047540      ;      CMP      #47540,R3      ;VERIFY FPP STATUS
12480 061546 001403             ;      BEQ     1$      ;BRANCH IF GOOD
12481 061550 004737 140132      ;      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12482 061554 104003             ;      ERROR   +3      ;FPP ERROR
12483             ;BAD FPP STSUS
12484 061556 022704 003254      1$:      CMP      #TAB3,R4      ;VERIFY AUTO-DEC
12485 061562 001403             ;      BEQ     2$      ;BRANCH IF GOOD
12486 061564 004737 140132      ;      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12487 061570 104003             ;      ERROR   +3      ;FPP ERROR
12488             ;BAD AUTO-INCR
12489 061572 012704 003254      2$:      MOV      #TAB3,R4      ;POINT TO RECEIVED DATA
12490 061576 004767 056276      ;      JSR     R7,DATVER      ;VERIFY DATA FROM FPP
12491 061602 005767 121312      ;      TST     COUNT      ;SEE IF COUNTER=0
12492 061606 001403             ;      BEQ     3$      ;BRANCH IF GOOD COMPARE
12493 061610 004737 140132      ;      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12494 061614 104003             ;      ERROR   +3      ;FPP ERROR
12495             ;BAD DATA FROM FPP
12496 061616      3$:
12497             ;
12500             ;
12501             ;-----
12502             ;TEST LDD MODE 5
12503             ;
12504 061616      ;MLDDM5:
12505             ;
12506 061616 012701 003142      ;      MOV      #RECDST,R1      ;POINT TO RECEIVED DATA LOCATION
12507 061622 012704 003164      ;      MOV      #TSTLOC+2,R4      ;POINT TO GOOD DATA
12508 061626 012737 003234 003162 ;      MOV      #TAB1,@TSTLOC      ;SET UP MODE 5 POINTER TO DATA
12509 061634 012702 047750      ;      MOV      #47750,R2      ;LOAD GOOD STATUS
12510 061640 170102             ;      LDFPS   R2      ;LOAD FPP STATUS - DOUBLE, ID
12511 061642 172454             ;      LDD      @-(R4),ACO      ;*TEST INSTRUCTION - MODE 5
12512 061644 170203             ;      STFPS   R3      ;SAVE TEST FPP STATUS
12513 061646 174011             ;      STD     ACO,(R1)      ;SAVE TEST RESULT
12514 061650 020203             ;      CMP      R2,R3      ;VERIFY FPP STATUS
12515 061652 001403             ;      BEQ     1$      ;BRANCH IF GOOD
12516 061654 004737 140132      ;      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12517 061660 104003             ;      ERROR   +3      ;FPP ERROR

```

FLOATING POINT TESTS

```

12518
12519 061662 022704 003162      1$:   CMP      @TSTLOC,R4      ;BAD FPP STATUS
12520 061666 001403              BEQ      2$                ;VERFIY AUTO-DEC
12521 061670 004737 140132      CALL     @@DETFPA          ;BRANCH IF GOOD
12522 061674 104003              ERROR    +3                ;DETERMINE FLOATING POINT FAULT. $$$
12523                                ;FPP ERROR
12524 061676 012704 003234      2$:   MOV      @TAB1,R4      ;BAD AUTO-DEC
12525 061702 004767 056172      JSR     R7,DATVER          ;POINT TO EXPECTED DATA
12526 061706 005767 121206      TST     COUNT              ;VERFIY DATA FROM FPP
12527 061712 001403              BEQ      3$                ;SEE IF COUNTER=0
12528 061714 004737 140132      CALL     @@DETFPA          ;BRANCH IF GOOD COMPARE
12529 061720 104003              ERROR    +3                ;DETERMINE FLOATING POINT FAULT. $$$
12530                                ;FPP ERROR
12531 061722                                ;BAD DATA FROM FPP
12534                                ;
12535                                ;-----
12536                                ;TEST LDD MODE 6
12537                                ;
12538 061722                                ;MLDDM6:
12539                                ;
12540 061722 012701 003342      MOV     @RECDST+200,R1     ;POINT TO RECEIVED DATA LOCATION
12541 061726 012704 003044      MOV     @TAB2-200,R4      ;SETUP R4 FOR MODE 6
12542 061732 012702 047750      MOV     @47750,R2         ;LOAD GOOD STATUS
12543 061736 170102              LDFPS   R2                ;LOAD FPP STATUS - DOUBLE.ID
12544 061740 172464 000200      LDD     200(R4),ACO       ;LDD MODE 6
12545 061744 170203              STFPS   R3                ;SAVE TEST FPP STATUS
12546 061746 174061 177600      STD     ACO,-200(R1)      ;SAVE TEST RESULT
12547 061752 022703 047740      CMP     @47740,R3        ;VERIFY FPP STATUS
12548 061756 001407              BEQ     1$                ;BRANCH IF GOOD
12549 061760 00473  140132      CALL     @@DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
12550 061764 104003              ERROR    +3                ;FPP ERROR?
12551                                ;BAD FPP STATUS
12552 061766 162701 000200      1$:   SUB     @200,R1        ;R1=RECDST
12553 061772 062704 000200      2$:   ADD     @200,R4        ;POINT TO EXPECTED DATA
12554 061776 004767 056076      JSR     R7,DATVER          ;VERFIY DATA FROM FPP
12555 062002 005767 121112      TST     COUNT              ;SEE IF COUNTER=0
12556 062006 001403              BEQ     3$                ;BRANCH IF GOOD COMPARE
12557 062010 004737 140132      CALL     @@DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
12558 062014 104003              ERROR    +3                ;FPP ERROR
12559                                ;BAD DATA FROM FPP
12560 062016                                ;
12563                                ;
12564                                ;-----
12565                                ;TEST LDD MODE 7
12566                                ;
12567 062016                                ;MLDDM7:
12568                                ;
12569 062016 012701 003142      MOV     @RECDST,R1        ;POINT TO RECEIVED DATA LOCATION
12570 062022 005004              CLR     R4                ;R4=0
12571 062024 012727 003234 003162  MOV     @TAB1,@TSTLOC     ;POINIER FOR MODE 7 GOOD DATA
12572 062032 012702 047750      MOV     @47750,R2         ;LOAD GOOD STATUS
12573 062036 170102              LDFPS   R2                ;LOAD FPP STATUS DOUBLE.ID
12574 062040 172474 003162      LDD     @TSTLOC(R4),ACO   ;*TEST INSTRUCTION - MODE 7
12575 062044 170203              STFPS   R3                ;SAVE TEST FPP STATUS
12576 062046 174011              STD     ACO.(R1)          ;SAVE TEST RESULT
12577 062050 020203              CMP     R2,R3             ;VERIFY FPP STATUS
12578 062052 001403              BEQ     1$                ;BRANCH IF GOOD

```

FLOATING POINT TESTS

```

12579 062054 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12580 062060 104003              ERROR   +3        ; FPP ERROR
12581                               ; BAD FPP STATUS
12582 062062 005704      1$:  TST   R4        ; VERIFY CONTENTS OF R4
12583 062064 001403      BEQ   2$        ; BRANCH IF GOOD
12584 062066 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12585 062072 104003              ERROR   +3        ; FPP ERROR
12586                               ; BAD R4
12587 062074 012704 003234      2$:  MOV   @TAB1,R4    ; POINT TO RECEIVED DATA
12588 062100 004767 055774      JSR   R7,DATVER   ; VERIFY DATA FROM FPP
12589 062104 005767 121010      TST   COUNT      ; SEE IF COUNTER=0
12590 062110 001403      BEQ   3$        ; BRANCH IF GOOD COMPARE
12591 062112 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12592 062116 104003              ERROR   +3        ; FPP ERROR
12593                               ; BAD DATA FROM FPP
12594 062120      3$:
12597                               ;
12598                               ;
12599                               ;-----
12600                               ; TEST LDD MODE 27 - ONLY 16 BITS ARE LOADED OR STORED
12601 062120      ; MLDM27:
12602                               ;
12603 062120 012701 003142      MOV   @RECDST,R1  ; POINT TO RECEIVED DATA LOCATION
12604 062124 012704 003274      MOV   @TAB5,R4    ; POINT TO GOOD D
12605 062130 012702 047750      MOV   @47750,R2   ; LOAD GOOD STATUS
12606 062134 005005      CLR   R5          ; R5=0
12607 062136 170102      LDFPS R2          ; LOAD FPP STATUS - DOUBLE.ID
12608 062140 172427 043243      LDD   @5205,ACO   ; *TEST INSTRUCTION - MODE 27
12609 062144 005205      INC   R5
12610 062146 005205      INC   R5
12611 062150 005205      INC   R5          ; TEST PROPER PC PATH
12612 062152 022705 000003      CMP   @3,R5       ; VERIFY ONLY 3 PC INCREMENT
12613 062156 001403      BEQ   1$        ; BRANCH IF PROPER PC ACTION
12614 062160 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12615 062164 104003              ERROR   +3        ; FPP ERROR
12616                               ; BAD MODE 27 LOAD
12617 062166 170203      1$:  STFPS R3          ; SAVE TEST FPP STATUS
12618 062170 174011      STD   ACO,(R1)    ; SAVE TEST RESULT
12619 062172 022703 047740      CMP   @47740,R3   ; VERIFY FPP STATUS
12620 062176 001403      BEQ   2$        ; BRANCH IF GOOD
12621 062200 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12622 062204 104003              ERROR   +3        ; FPP ERROR
12623                               ; BAD FPP STATUS
12624 062206 004767 055666      2$:  JSR   R7,DATVER   ; VERIFY DATA FROM FPP
12625 062212 005767 120702      TST   COUNT      ; SEE IF COUNTER=0
12626 062216 001403      BEQ   3$        ; BRANCH IF GOOD COMPARE
12627 062220 004737 140132      CALL  @DEFPA      ; DETERMINE FLOATING POINT FAULT. $$$
12628 062224 104003              ERROR   +3        ; FPP ERROR
12629                               ; BAD DATA FROM FPP
12630 062226      3$:
12633                               ;
12634                               ;
12635                               ;-----
12636                               ; TEST ADDF, ADDD, SUBF, SUBD - ACO=0 FSRC=0;
12637 062226      ; MNRM1:
12638                               ;
12639 062226 012704 003314      MOV   @TAB6,R4    ; POINT TO FSRC TEST DATA

```

FLOATING POINT TESTS

12640	062232	005067	120710		CLR	RECDST+4		;CLEAR OUT RECEIVED DATA TABLE
12641	062236	005067	120706		CLR	RECDST+6		;
12642	062242	012702	040000		MOV	#40000,R2		;SET UP GOOD STATUS
12643	062246	170102			LDFPS	R2		;LOAD FPP STATUS, FLOATING
12644	062250	172414			LDF	(R4),ACO		;LOAD ACO WITH 0
12645	062252	172014			ADDF	(R4),ACO		;0+0
12646	062254	170203			STFPS	R3		;SAVE STATUS
12647	062256	022703	040004		CMP	#40004,R3		;VERIFY STATUS
12648	062262	001403			BEQ	1\$;BRANCH IF GOOD
12649	062264	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12650	062270	104003			ERROR	+3		;FPP ERROR
12651								;BAD FPP STATUS
12652	062272	012701	003142	1\$:	MOV	#RECDST,R1		;POINT TO RECEIVERD DATA
12653	062276	174011			STF	ACO,(R1)		;SAVE DATA
12654	062300	004767	055574		JSR	R7,DATVER		;VERIFY DATA
12655	062304	005767	120610		TST	COUNT		;
12656	062310	001403			BEQ	2\$;BRANCH IF GOOD
12657	062312	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12658	062316	104003			ERROR	+3		;FPP ERROR
12659								;BAD DATA IN ACO
12660	062320	012702	040200	2\$:	MOV	#40200,R2		;LOAD FLOATING STATUS
12661	062324	170102			LDFPS	R2		;
12662	062326	172414			LDD	(R4),ACO		;LOAD ACO WITH 0
12663	062330	172014			ADDD	(R4),ACO		;*TEST INSTRUCTION
12664	062332	174011			STD	ACO,(R1)		;SAVE DATA
12665	062334	170203			STFPS	R3		;SAVE FPS
12666	062336	022703	040204		CMP	#40204,R3		;VERIFY STATUS
12667	062342	001403			BEQ	3\$;BRANCH IF GOOD
12668	062344	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12669	062350	104003			ERROR	+3		;FPP ERROR
12670								;BAD FPS
12671	062352	004767	055522	3\$:	JSR	R7,DATVER		;VERIFY DATA
12672	062356	005737	003120		TST	#COUNT		;VERIFY RESULT
12673	062362	001403			BEQ	44\$;BRANCH IF GOOD
12674	062364	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12675	062370	104003			ERROR	+3		;FPP ERROR
12676								;BAD ACO
12677	062372	172414		44\$:	LDD	(R4),ACO		;SETUP DATA
12678	062374	173014			SUBD	(R4),ACO		;*TEST INSTRUCTION
12679	062376	170203			STFPS	R3		;SAVE STATUS
12680	062400	022703	040204		CMP	#40204,R3		;VERIFY STATUS
12681	062404	001403			BEQ	4\$;BRANCH IF GOOD
12682	062406	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12683	062412	104003			ERROR	+3		;FPP ERROR
12684								;BAD FPS
12685	062414	174011		4\$:	STD	ACO,(R1)		;SAVE ACO DATA
12686	062416	004767	055456		JSR	R7,DATVER		;VERIFY DATA
12687	062422	005767	120472		TST	COUNT		;
12688	062426	001403			BEQ	5\$;BRANCH IF GOOD
12689	062430	004737	140132		CALL	#DEFPA		;DETERMINE FLOATING POINT FAULT. \$\$\$
12690	062434	104003			ERROR	+3		;FPP ERROR
12691								;BAD ACO
12692	062436	170127	000000	5\$:	LDFPS	#0		;STORE FPP STATUS
12693	062442	172414			LDD	(R4),ACO		;LOAD ACO
12694	062444	173014			SUBF	(R4),ACO		;0-0
12695	062446	170203			STFPS	R3		;SAVE STATUS
12696	062450	174011			STD	ACO,(R1)		;SAVE ACO

FLOATING POINT TESTS

```

12697 062452 022703 000004      CMP      #4,R3          ;VERFIY STATUS
12698 062456 001403              BEQ      6$            ;BRANCH IF GOOD
12699 062460 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12700 062464 104003              ERROR   +3            ;FPP ERROR
12701                                ;BAD FPS
12702 062466 004767 055406      6$:   JSR      R7,DATVER ;VERIFY DATAT
12703 062472 005767 120422      TST     COUNT        ;
12704 062476 001403              BEQ      7$            ;BRANC IF GOOD
12705 062500 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12706 062504 104003              ERROR   +3            ;FPP ERROR
12707                                ;BAD ACO
12708 062506
12709                                ;
12710                                ;
12711                                ;
12712                                ;
12713                                ;
12714                                ;TEST ADDF,SUBD - FSRC=0, ACO NE 0
12715                                ;
12716 062506      MNNRM2:
12717                                ;
12718 062506 012701 003142      MOV     #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
12719 062512 012705 003314      MOV     #TAB6,R5      ;POINT TO SOURCE DATA TABLE
12720 062516 012704 003244      MOV     #TAB2,R4      ;POINT TO ACO DATA
12721 062522 170127 000200      LDFPS  #200           ;SET TO DOUBLE FOR CLEAR
12722 062526 172415              LDD     (R5),ACO      ;
12723 062530 005002              CLR     R2            ;SETUP FPP STATUS
12724 062532 170102      LDFPS  R2            ;LOAD FPS
12725 062534 172414      LDF     (R4),ACO      ;LOAD ACO
12726 062536 172015      ADDF   (R5),ACO      ;*TEST INSTRUCTION
12727 062540 170203      STFPS  R3            ;SAVE STATUS
12728 062542 174011      STF     ACO,(R1)      ;SAVE ACO
12729 062544 022703 000000      CMP     #0,R3         ;VERFIY NEGATIVE RESULT
12730 062550 001403              BEQ     1$            ;BRANCH IF GOOD
12731 062552 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12732 062556 104003              ERROR   +3            ;FPP ERROR
12733                                ;BAD FPS
12734 062560 012704 003264      1$:   MOV     #TAB4,R4      ;POINT TO EXPECTED DATA
12735 062564 004767 055310      JSR     R7,DATVER     ;VERFIY ACO
12736 062570 005767 120324      TST     COUNT        ;CHECK RESULT
12737 062574 001403              BEQ     2$            ;BRANCH IF GOOD
12738 062576 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12739 062602 104003              ERROR   +3            ;FPP ERROR
12740                                ;BAD ACO
12741 062604 170127 000200      2$:   LDFPS  #200           ;SET STATUS TO DUOBLE NODE
12742 062610 172414      LDD     (R4),ACO      ;LOAD ACO WITH A VALUE
12743 062612 173015      SUBD   (R5),ACO      ;*TEST INSTRUCTION
12744 062614 170203      STFPS  R3            ;SAVE FPP STATUS
12745 062616 174011      STD     ACO,(R1)      ;SAVE ACO
12746 062620 022703 000200      CMP     #200,R3       ;VERIFY RESULT
12747 062624 001403              BEQ     3$            ;BRANCH IF GOOD
12748 062626 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
12749 062632 104003              ERROR   +3            ;FPP ERROR
12750                                ;BAD SUBD
12751 062634 012704 003264      3$:   MOV     #TAB4,R4      ;POINT TO EXPECTED
12752 062640 004767 055234      JSR     R7,DATVER     ;VERIFY ACO
12753 062644 005767 120250      TST     COUNT        ;
12754 062650 001403              BEQ     4$            ;BRANCH IF GOOD ACO
12755 062652 004737 140132      CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$

```

FLOATING POINT TESTS

```

12756 062656 104003          ERROR +3          ;FPP ERROR
12757                          ;BAD ACO
12758 062660          4$:
12761                          ;
12762                          ;-----
12763                          ;TEST ADDD, SUBF - FSRC NE 0, ACO=0
12764                          ;
12765 062660          MNNRM3:
12766                          ;
12767 062660 012701 003142      MOV      #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12768 062664 012705 003314      MOV      #TAB6,R5       ;POINT TO ACO DATA TABLE
12769 062670 012704 003234      MOV      #TAB1,R4      ;POINT TO FSRC DATA
12770 062674 012702 000200      MOV      #200,R2       ;SETUP FPP STATUS
12771 062700 170102              LDFPS   R2              ;LOAD FPS
12772 062702 172415              LDD     (R5),ACO       ;LOAD ACO
12773 062704 172014              ADDD   (R4),ACO       ;*TEST INSTRUCTION
12774 062706 170203              STFPS  R3              ;SAVE STATUS
12775 062710 174011              STD    ACO,(R1)       ;SAVE ACO
12776 062712 022703 000210      CMP     #210,R3       ;VERFIY NEGATIVE RESULT
12777 062716 001403              BEQ    1$             ;BRANCH IF GOOD
12778 062720 004737 140132      CALL   @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12779 062724 104003          ERROR +3          ;FPP ERROR
12780                          ;BAD FPS
12781 062726 004767 055146      1$:      JSR     R7,DATVER      ;VERFIY ACO
12782 062732 005767 120162      TST    COUNT         ;CHECK RESULT
12783 062736 001403              BEQ    2$             ;BRANCH IF GOOD
12784 062740 004737 140132      CALL   @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
12785 062744 104003          ERROR +3          ;FPP ERROR
12786                          ;BAD ACO
12787 062746 170127 000200      2$:      LDFPS  #200           ;SET STATUS TO DUOBLE NODE
12788 062752 172415              LDF    (R5),ACO      ;LOAD ACO WITH A VALUE
12789 062754 173014              SUBD   (R4),ACO      ;*TEST INSTRUCTION
12790 062756 170203              STFPS  R3           ;SAVE FPP STATUS
12791 062760 174011              STF    ACO,(R1)     ;SAVE ACO
12792 062762 022703 000200      CMP     #200,R3     ;VERIFY RESULT
12793 062766 001403              BEQ    3$           ;BRANCH IF GOOD
12794 062770 004737 140132      CALL   @DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
12795 062774 104003          ERROR +3          ;FPP ERROR
12796                          ;BAD SUBD
12797 062776 012704 003504      3$:      MOV     #TAB18,R4    ;POINT TO EXPECTED DATA
12798 063002 004767 055072      JSR    R7,DATVER    ;VERIFY ACO
12799 063006 005767 120106      TST    COUNT       ;
12800 063012 001403              BEQ    4$           ;BRANCH IF GOOD ACO
12801 063014 004737 140132      CALL   @DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
12802 063020 104003          ERROR +3          ;FPP ERROR
12803                          ;BAD ACO
12804 063022          4$:
12805                          ;
12808                          ;
12809                          ;-----
12810                          ;TEST ADDF, SUBD - EXP(ACO) = EXP(FSRC)
12811                          ;
12812 063022          MNNRM4:
12813                          ;
12814 063022 012702 003240      MOV     #3240,R2     ;SET FIU,FD,FT
12815 063026 170102              LDFPS  R2
12816 063030 012704 003334      MOV     #TAB7,R4     ;SET FSRC

```

FLOATING POINT TESTS

```

12817 063034 012705 003344      MOV      #TAB8,R5          ;SETUP ACO
12818 063040 012701 003142      MOV      #RECDST,R1       ;POINT TO RECEIVED DATA
12819 063044 172415              LDD      (R5),ACO         ;LOAD ACO
12820 063046 172014              ADDD     (R4),ACO         ;*TEST INSTRUCTION
12821 063050 174011              STD      ACO,(R1)        ;SAVE TEST RESULT
12822 063052 012704 003354      MOV      #TAB9,R4        ;POINT TO EXPECTED DATA
12823 063056 004767 055016      JSR      R7,DATVER       ;VERIFY ACO DATA
12824 063062 005767 120032      TST      COUNT          ;
12825 063066 001403              BEQ      1$              ;BRANCH IF GOOD
12826 063070 004737 140132      CALL     @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12827 063074 104003              ERROR    +3              ;FPP ERROR
12828                               ;BAD ADDO
12829 063076 012704 003344      1$:  MOV      #TAB8,R4          ;
12830 063102 012703 003344      MOV      #TAB8,R3          ;SETUP SAME ACO
12831 063106 012702 003200      MOV      #3200,R2         ;ROUND MODE
12832 063112 170102              LDFPS   R2
12833 063114 172413              LDD      (R3),ACO         ;LOAD ACO
12834 063116 061400              ADD      (R4),ACO         ;*TEST INSTRUCTION
12835 063120 174011              STD      ACO,(R1)        ;SAVE DATA
12836 063122 004767 054752      JSR      R7,DATVER       ;VERIFY ACO
12837 063126 005767 117766      TST      COUNT          ;
12838 063132 001403              BEQ      2$              ;BRANCH IF GOOD
12839 063134 004737 140132      CALL     @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12840 063140 104003              ERROR    +3              ;FPP ERROR
12841                               ;BAD ROUND RESULT
12842 063142              2$:
12843              ;
12844              ;
12845              ;-----
12846              ;TEST ADD - EXP(FSRC) > EXP(ACO)
12847              ;
12848              ;
12849              ;
12850 063142              MXDF1:
12851              ;
12852 063142 012702 003200      MOV      #3200,R2         ;R2-FPP STATUS
12853 063146 170102              LDFPS   R2               ;LOAD FPS STATUS
12854 063150 012704 003374      MOV      #TAB11,R4        ;POINT TO FSRC DATA
12855 063154 012701 003142      MOV      #RECDST,R1       ;POINT TO ACO RESULT
12856 063160 012705 003364      MOV      #TAB10,R5        ;POINT TO ACO DATA
12857 063164 172415              LDD      (R5),ACO         ;LOAD ACO DATA
12858 063166 172014              ADDD     (R4),ACO         ;*TEST INSTRUCTIONS
12859 063170 170203              STFPS   R3               ;SAVE FPP STATUS
12860 063172 174011              STD      ACO,(R1)        ;SAVE ACO DATA
12861 063174 022703 003200      CMP      #3200,R3         ;VERIFY FPP STATUS
12862 063200 001403              BEQ      1$              ;BRANCH IF GOOD
12863 063202 004737 140132      CALL     @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12864 063206 104003              ERROR    +3              ;FPP ERROR
12865                               ;BAD FPP STATUS
12866 063210 012704 003404      1$:  MOV      #TAB11A,R4       ;POINT TO EXPECTED DATA
12867 063214 004767 054660      JSR      R7,DATVER       ;VERIFY CONTENTS OF ACO
12868 063220 005767 117674      TST      COUNT          ;
12869 063224 001403              BEQ      2$              ;BRANCH IF GOOD ACO
12870 063226 004737 140132      CALL     @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12871 063232 104003              ERROR    +3              ;FPP ERROR
12872                               ;BAD ACO, SHOULD = FSRC
12873 063234 012704 003414      2$:  MOV      #TAB12,R4        ;POINT TO FSRC DATA
12874 063240 172415              LDD      (R5),ACO         ;ACO
12875 063242 172014              ADDD     (R4),ACO         ;*TEST INSTRUCTION

```

FLOATING POINT TESTS

```

12876 063244 012704 003434      MOV      #TAB13B,R4      ;POINT TO EXPECTED RESULT
12877 063250 174011             STD      ACO,(R1)       ;SAVE ACO DATA INTO RECDAT
12878 063252 004767 054622      JSR      R7,DATVER      ;VERIFY DATA
12879 063256 005767 117636      TST      COUNT
12880 063262 001403             BEQ      3$             ;BRANCH IF GOOD DATA
12881 063264 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12882 063270 104003             ERROR    +3            ;FPP ERROR
12883                                     ;BAD ACO DATA
12884 063272 012702 003000      3$:    MOV      #3000,R2      ;GET FPP STATUS DATA
12885 063276 012704 003444      MOV      #TAB14,R4      ;POINT TO FSRC DATA
12886 063302 012705 003454      MOV      #TAB15,R5      ;POINT TO ACO DATA
12887 063306 172415             LDD      (R5),ACO       ;LOAD ACO
12888 063310 170102             LDFPS   R2              ;FPP STATUS = FLOAT, INTERRUPTS ENABLE
12889 063312 172014             ADDF    (R4),ACO        ;*TEST INSTRUCTION
12890 063314 170127 000200      LDFPS   #200           ;RESET TO DOUBLE
12891 063320 174011             STD      ACO,(R1)       ;RECDST=ACO
12892 063322 012704 003364      MOV      #TAB10,R4      ;POINT TO GOOD DATA
12893 063326 004767 054546      JSR      R7,DATVER      ;VERFIY CONTENTS OF ACO
12894 063332 005767 117562      TST      COUNT
12895 063336 001403             BEQ      4$             ;BRANCH IF GOOD
12896 063340 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12897 063344 104003             ERROR    +3            ;FPP ERROR
12898                                     ;BAD FLOATING ADD
12899 063346 012705 003464      4$:    MOV      #TAB16,R5      ;POINT TO ACO DATA
12900 063352 170102             LDFPS   R2              ;FPP STATUS = FLOAT
12901 063354 172415             LDF     (R5),ACO        ;LOAD ACO
12902 063356 172014             ADDF    (R4),ACO        ;*TEST INSTRUCTION
12903 063360 174011             STD      ACO,(R1)       ;SAVE ACO DATA
12904 063362 012704 003474      MOV      #TAB17,R4      ;POINT TO GOOD DATA
12905 063366 004767 054506      JSR      R7,DATVER
12906 063372 005767 117522      TST      COUNT
12907 063376 001403             BEQ      5$             ;BRANCH IF GOOD
12908 063400 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12909 063404 104003             ERROR    +3            ;FPP ERROR
12910                                     ;BAD FLOATING ADD
12911 063406             5$:
12912             ;
12915             ;
12916             ;
12917             ;-----
12918             ;TEST ADD WITH NEGATIVE OPERANDS
12919 063406             ;
12920             ;MNGOP:
12921 063406 012702 003200      ;
12922 063412 170102             MOV      #3200,R2      ;LOAD FPS VALUE
12923 063414 012704 003514      LDFPS   R2              ;
12924 063420 172414             MOV      #TAB21,R4      ;DATA ADDRESS FOR ACO AND FSR
12925 063422 172014             LDD      (R4),ACO       ;ACO=100200 0 0 0
12926 063424 170203             ADDD    (R4),ACO       ;*TEST INSTRUCTION
12927 063426 012701 003142      STFPS   R3              ;SAVE STATUS
12928 063432 174011             MOV      #RECDST,R1     ;POINT TO RECEIVED DATA TABLE
12929 063434 022703 003210      STD      ACO,(R1)       ;SAVE ACO DATA
12930 063440 001403             CMP      #3210,R3       ;VERIFY STATUS
12931 063442 004737 140132      BEQ      1$             ;BRANCH IF GOOD
12932 063446 104003             CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12933                                     ;FPP ERROR
12934 063450 012704 003524      1$:    MOV      #TAB22,R4      ;POINT TO EXPECTED DATA

```


FLOATING POINT TESTS

```

12935 063454 004767 054420      JSR      R7,DATVER      ;
12936 063460 005767 117434      TST      COUNT          ;VERIFY DATA
12937 063464 001403              BEQ      2$             ;BRANCH IF GOOD
12938 063466 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12939 063472 104003              ERROR    +3            ;FPP ERROR
12940
12941
12942 063474 012704 003514      2$:     MOV      @TAB21,R4      ;POINT TO FSRC DATA
12943 063500 012701 003534      MOV      @TAB23,R1      ;POINT TO ACO DATA
12944 063504 012737 063526 000244  MOV      @101$,@FPVEC    ;SETUP FP VECTOR
12945 063512 172411              LDD      (R1),ACO        ;LOAD ACO
12946 063514 172014              ADDD     (R4),ACO        ;*TEST INSTRUCTION
12947 063516 170000              100$:   CFCC              ;COPY FPP CC
12948 063520 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12949 063524 104003              ERROR    +3            ;FPP ERROR
12950
12951 063526 170203              101$:   STFPS     R3          ;SAVE FPP STATUS
12952 063530 012701 003142      MOV      @RECDST,R1      ;POINT TO RECEIVED DATA TABLE
12953 063534 174011              STD      ACO,(R1)        ;SAVE ACO DATA
12954 063536 022703 103200      CMP      @103200,R3      ;VERIFY STATUS
12955 063542 001403              BEQ      3$             ;BRANCH IF GOOD
12956 063544 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12957 063550 104003              ERROR    +3            ;FPP ERROR
12958
12959 063552 012605              3$:     MOV      (SP)+,R5      ;GET ERROR PC
12960 063554 020527 063516      CMP      R5,@100$        ;VERIFY ERROR ADDRESS ON STACK
12961 063560 001403              BEQ      102$           ;BRANCH IF GOOD
12962 063562 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12963 063566 104003              ERROR    +3            ;FPP ERROR
12964
12965 063570 005726              102$:   TST      (SP)+        ;RESTORE STACK
12966 063572 012704 003544      MOV      @TAB24,R4      ;POINT TO EXPECTED DATA TABLE
12967 063576 004767 054276      JSR      R7,DATVER      ;VERIFY DATA
12968 063602 005767 117312      TST      COUNT          ;BRANC IF GOOD
12969 063606 001403              BEQ      4$             ;DETERMINE FLOATING POINT FAULT. $$$
12970 063610 004737 140132      CALL     @DETFPA        ;FPP ERROR
12971 063614 104003              ERROR    +3            ;BAD ACO DATA
12972
12973
12974 063616 012704 003534      4$:     MOV      @TAB23,R4      ;POINT TO FSRC DATA
12975 063622 012701 003514      MOV      @TAB21,R1      ;POINT TO ACO DATA
12976 063626 012737 063656 000244  MOV      @104$,@FPVEC    ;SETUP FP VECTOR
12977 063634 012702 003200      MOV      @3200,R2        ;LOAD FPS VALUE
12978 063640 170102              LDFPS   R2              ;
12979 063642 172411              LDD      (R1),ACO        ;LOAD ACO DATA
12980 063644 172014              ADDD     (R4),ACO        ;*TEST INSTRUCTION
12981 063646 170000              103$:   CFCC              ;COPY FPP CC
12982 063650 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12983 063654 104003              ERROR    +3            ;FPP ERROR
12984
12985 063656 170203              104$:   STFPS     R3          ;SAVE FPS
12986 063660 012701 003142      MOV      @RECDST,R1      ;SAVE ACO
12987 063664 174011              STD      ACO,(R1)        ;
12988 063666 022703 103200      CMP      @103200,R3      ;VERFIY STATUS
12989 063672 001403              BEQ      5$             ;BRANCH IF GOOD
12990 063674 004737 140132      CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
12991 063700 104003              ERROR    +3            ;FPP ERROR

```

FLOATING POINT TESTS

```

12992
12993 063702 012605
12994 063704 020527 063646
12995 063710 001403
12996 063712 004737 140132
12997 063716 104003
12998
12999 063720 005726
13000 063722 012704 003544
13001 063726 004767 054146
13002 063732 005767 117162
13003 063736 001403
13004 063740 004737 140132
13005 063744 104003
13006
13007
13008 063746 012704 003564
13009 063752 012701 003554
13010 063756 012702 003200
13011 063762 170102
13012 063764 012737 000246 000244
13013 063772 172411
13014 063774 172014
13015 063776 170203
13016 064000 012701 003142
13017 064004 174011
13018 064006 020327 003200
13019 064012 001403
13020 064014 004737 140132
13021 064020 104003
13022
13023 064022 012704 003574
13024 064026 004767 054046
13025 064032 005767 117062
13026 064036 001403
13027 064040 004737 140132
13028 064044 104003
13029
13030
13031 064046 012704 003554
13032 064052 012701 003564
13033 064056 172411
13034 064060 172014
13035 064062 170203
13036 064064 012701 003142
13037 064070 174011
13038 064072 020327 003200
13039 064076 001403
13040 064100 004737 140132
13041 064104 104003
13042
13043 064106 012704 003574
13044 064112 004767 053762
13045 064116 005767 116776
13046 064122 001403
13047 064124 004737 140132
13048 064130 104003

5$: MOV (SP)+,R5 ;BAD FPS STATUS
CMP R5,#103$ ;GET ERROR PC
BEQ 105$ ;VERIFY ERROR ADDRESS ON STACK
CALL @DEFPA ;BRANCH IF GOOD
ERROR +3 ;DETERMINE FLOATING POINT FAULT. $$$
;FPP ERROR
;BAD ERROR RETURN ON STACK
;RESTORE STACK
;POINT TO EXPECTED DATA

105$: TST (SP)+
MOV @TAB24,R4
JSR R7,DATVER
TST COUNT
BEQ 6$ ;BRANCH IF GOOD
CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;BAD ACO
;ACO!

6$: ; !-FSRC! <
MOV @TAB26,R4 ;POINT TO FSRC DATA
MOV @TAB25,R1 ;POINT TO ACO DATA
MOV @3200,R2 ;LOAD FPS VALUE
LDFPS R2
MOV @246,@FPVEC ;SETUP FP VECTOR
LDD (R1),ACO ;LOAD ACO DATA
ADD (R4),ACO ;*TEST INSTRUCTION
STFPS R3 ;SAVE STATUS
MOV @RECDST,R1 ;POINT TO RECEIVED DATA TABLE
STD ACO,(R1) ;SAVE ACO
CMP R3,#3200 ;VERIFY STATUS
BEQ 7$ ;BRANCH IF GOOD
CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;BAD FPS

7$: MOV @TAB27,R4 ;POINT TO EXPECTED DSATA
JSR R7,DATVER ;VERIFY DATA
TST COUNT
BEQ 8$ ;BRANCH IF GOOD
CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR

; !-AC!
8$: ; !FSRC! >
MOV @TAB25,R4 ;POINT TO FSRC DATA
MOV @TAB26,R1 ;POINT TO ACO DATA
LDD (R1),ACO ;LOAD ACO DATA
ADD (R4),ACO ;*TEST INSTRUCTION
STFPS R3 ;SAVE STATUS
MOV @RECDST,R1 ;POINT TO RECEIVED DATA TABLE
STD ACO,(R1) ;SAVE ACO
CMP R3,#3200 ;VERIFY STATUS
BEQ 9$ ;BRANCH IF GOOD
CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;BAD FPS

9$: MOV @TAB27,R4 ;POINT TO EXPECTED DSATA
JSR R7,DATVER ;VERIFY DATA
TST COUNT
BEQ 10$ ;BRANCH IF GOOD
CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR

```

FLOATING POINT TESTS

```

13049                                     ;BAD ACO
13050                                     !ACO!
13051 064132 012704 003614          10$:  MOV   #TAB29,R4           ;POINT TO FSRC DATA
13052 064136 012701 003604          MOV   #TAB28,R1           ;POINT TO ACO DATA
13053 064142 172411                  LDD   (R1),ACO           ;LOAD ACO DATA
13054 064144 172014                  ADDD  (R4),ACO           ;*TEST INSTRUCTION
13055 064146 170203                  STFPS R3                 ;SAVE STATUS
13056 064150 012701 003142          MOV   #RECDST,R1        ;POINT TO RECEIVED DATA TABLE
13057 064154 174011                  STD   ACO,(R1)           ;SAVE ACO
13058 064156 020327 003200          CMP   R3,#3200          ;VERIFY STATUS
13059 064162 001403                  BEQ   11$                ;BRANCH IF GOOD
13060 064164 004737 140132          CALL  @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13061 064170 104003                  ERROR +3                 ;FPP ERROR
13062                                     ;BAD FPS
13063 064172 012704 003624          11$:  MOV   #TAB29A,R4      ;POINT TO EXPECTED DATA
13064 064176 004767 053676          JSR   R7,DATVER         ;VERIFY DATA
13065 064202 005767 116712          TST   COUNT
13066 064206 001403                  BEQ   12$                ;BRANCH IF GOOD
13067 064210 004737 140132          CALL  @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13068 064214 104003                  ERROR +3                 ;FPP ERROR
13069                                     ;
13070 064216          12$:
13071                                     ;
13074                                     ;
13075                                     ;
13076                                     ;TEST SUB WITH EXP[ACO]=EXP[FSRC]
13077                                     ;
13078 064216          MSB:
13079                                     ;
13080 064216 012702 003200          MOV   #3200,R2           ;LOAD FPS DATA
13081 064222 170102                  LDFPS R2                 ;LOAD FPS
13082 064224 012704 003514          MOV   #TAB21,R4         ;POINT TO FSRC DATA
13083 064230 012701 003142          MOV   #RECDST,R1        ;POINT TO ACO RECEIVED DATA TABLE
13084 064234 172414                  LDD   (R4),ACO           ;LOAD ACO
13085 064236 173014                  SUBD  (R4),ACO           ;*TEST INSTRUCTION
13086 064240 170203                  STFPS R3                 ;SAVE STATUS
13087 064242 174011                  STD   ACO,(R1)           ;SAVE ACO INTO RECDST
13088 064244 022703 003204          CMP   #3204,R3          ;VERIFY STATUS
13089 064250 001403                  BEQ   1$                 ;BRANCH IF GOOD
13090 064252 004737 140132          CALL  @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13091 064256 104003                  ERROR +3                 ;FPP ERROR
13092                                     ;BAD FPS STATUS
13093 064260 012704 003314          1$:   MOV   #TAB6,R4       ;POINT TO EXPECTED DATA
13094 064264 004767 053610          JSR   R7,DATVER         ;VERIFY ACO
13095 064270 005767 116624          TST   COUNT
13096 064274 001403                  BEQ   2$                 ;BRANCH IF GOOD
13097 064276 004737 140132          CALL  @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13098 064302 104003                  ERROR +3                 ;FPP ERROR
13099                                     ;BAD ACO
13100 064304 012704 003444          2$:   MOV   #TAB14,R4      ;POINT TO FSRC AND ACO DATA
13101 064310 172414                  LDD   (R4),ACO           ;LOAD ACO DATA
13102 064312 173014                  SUBD  (R4),ACO           ;*TEST INSTRUCTION
13103 064314 170203                  STFPS R3                 ;SAVE FPS
13104 064316 174011                  STD   ACO,(R1)           ;SAVE ACO INTO RECDST
13105 064320 022703 003204          CMP   #3204,R3          ;VERIFY FPS
13106 064324 001403                  BEQ   3$                 ;BRANCH IF GOOD
13107 064326 004737 140132          CALL  @#DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$

```

FLOATING POINT TESTS

```

13108 064332 104003          ERROR   +3          ;FPP ERROR
13109                               ;BAD ACO
13110 064334 012704 003314    3$:   MOV    #TAB6,R4          ;POINT TO EXPECTED DATA
13111 064340 004767 053534          JSR    R7,DATVER          ;VERIFY ACO
13112 064344 005767 116550          TST    COUNT
13113 064350 001403          BEQ    4$                  ;BRANCH IF GOOD
13114 064352 004737 140132          CALL   @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13115 064356 104003          ERROR   +3          ;FPP ERROR
13116                               ;BAD ACO
13117 064360          4$:
13118                               ;
13121                               ;
13122                               ;
13123                               ;-----
13124                               ;TEST NORMALIZE
13125 064360          MNRM:
13126                               ;
13127 064360 012702 003200          MOV    #3200,R2          ;LOAD FPS
13128 064364 170102          LDFPS  R2
13129 064366 012705 003644          MOV    #TAB31,R5        ;POINT TO FSRC DATA
13130 064372 012701 003634          MOV    #TAB30,R1        ;POINT TO ACO DATA
13131 064376 172411          LDD    (R1),ACO         ;LOAD ACO
13132 064400 173015          SUBD   (R5),ACO         ;*TEST INSTRUCTION
13133                               ;1 LEFT SHIFT
13134 064402 170203          STFPS  R3               ;SAVE STATUS
13135 064404 012704 003142          MOV    #RECDST,R4       ;POINT TO RECDATA
13136 064410 174014          STD    ACO,(R4)         ;SAVE ACO
13137 064412 012701 003674          MOV    #TAB34,R1        ;POINT TO EXPECTED DATA
13138 064416 004767 053456          JSR    R7,DATVER          ;VERIFY DATA
13139 064422 005767 116472          TST    COUNT
13140 064426 001403          BEQ    1$                  ;BRANCH IF GOOD
13141 064430 004737 140132          CALL   @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13142 064434 104003          ERROR   +3          ;FPP ERROR
13143                               ;
13144 064436 012701 003654          1$:   MOV    #TAB32,R1          ;ACO DATA
13145 064442 012705 003664          MOV    #TAB33,R5        ;FSRC DATA
13146 064446 172411          LDD    (R1),ACO         ;LOAD ACO
13147 064450 173015          SUBD   (R5),ACO         ;*TST INSTRUCTION
13148                               ;56 LEFT SHIFTS
13149 064452 012701 003142          MOV    #RECDST,R1       ;SAVE DATA
13150 064456 174011          STD    ACO,(R1)
13151 064460 004767 053414          JSR    R7,DATVER
13152 064464 005767 116430          TST    COUNT
13153 064470 001403          BEQ    2$                  ;
13154 064472 004737 140132          CALL   @DETFPA          ;DETERMINE FLOATING POINT FAULT. $$$
13155 064476 104003          ERROR   +3          ;FPP ERROR
13156                               ;
13157 064500          2$:
13158                               ;
13161                               ;
13162                               ;
13163                               ;-----
13164                               ;TEST ADD WITH OVERFLOW AND UNDERFLOW
13165 064500          MUVAD:
13166                               ;
13167 064500 012702 000200          MOV    #200,R2          ;SETUP FLOATING POINT STATUS
13168 064504 170102          LDFPS  R2               ;LOAD FPS

```

FLOATING POINT TESTS

```

13169 064506 012704 003704      MOV      #TAB40,R4      ;POINT TO FSRC DATA
13170 064512 012701 003704      MOV      #TAB40,R1      ;POINT TO ACO DATA
13171 064516 172411              LDD      (R1),ACO      ;LOAD ACO WITH TEST DATA
13172 064520 172014              ADDD     (R4),ACO      ;*TEST INSTRUCTION
13173 064522 170203              STFPS    R3            ;SAVE FPS
13174 064524 012701 003142      MOV      #RECDST,R1     ;POINT TO RECEIVED DATA TABLE
13175 064530 174011              STD      ACO,(R1)      ;SAVE ACO RESULT
13176 064532 022703 000206      CMP      #206,R3       ;VERIFY STATUS
13177 064536 001403              BEQ      1$            ;BRANCH IF GOOD
13178 064540 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13179 064544 104003              ERROR    +3            ;FPP ERROR
13180                                ;BAD FPS
13181 064546 012704 003314      1$:     MOV      #TAB6,R4      ;POINT TO EXPECTED DATA
13182 064552 004767 053322      JSR      R7,DATVER     ;VERIFY DATA
13183 064556 005767 116336      TST      COUNT
13184 064562 001403              BEQ      2$            ;BRANCH IF GOOD
13185 064564 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13186 064570 104003              ERROR    +3            ;FPP ERROR
13187                                ;BAD ACO
13188                                ;OVERFLOW TRAPS ENABLED
13189                                ;
13190 064572 012702 001200      2$:     MOV      #1200,R2     ;SETUP FLOATING POINT STATUS
13191 064576 170102              LDFPS    R2            ;LOAD FPS
13192 064600 012704 003704      MOV      #TAB40,R4     ;POINT TO FSRC DATA
13193 064604 012701 003704      MOV      #TAB40,R1     ;POINT TO ACO DATA
13194 064610 172411              LDD      (R1),ACO      ;LOAD ACO WITH TEST DATA
13195 064612 012737 064632 000244  MOV      #3$,@FPVEC    ;CHANGE TRAP VECTOR
13196 064620 172014              ADDD     (R4),ACO      ;*TEST INSTRUCTION
13197 064622 170000              CFCC
13198 064624 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13199 064630 104003              ERROR    +3            ;FPP ERROR
13200                                ;FAILED TO TRAP ON OVERFLOW
13201 064632 170203              3$:     STFPS    R3            ;SAVE FPS
13202 064634 012701 003142      MOV      #RECDST,R1     ;POINT TO RECEIVED DATA TABLE
13203 064640 174011              STD      ACO,(R1)      ;SAVE ACO RESULT
13204 064642 022703 101206      CMP      #101206,R3    ;VERIFY STATUS
13205 064646 001403              BEQ      4$            ;BRANCH IF GOOD
13206 064650 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13207 064654 104003              ERROR    +3            ;FPP ERROR
13208                                ;BAD FPS
13209 064656 012600              4$:     MOV      (SP)+,R0     ;CHECK STORED PC
13210 064660 022700 064622      CMP      #23$,R0
13211 064664 001403              BEQ      5$            ;BRANCH IF RETURN ADDRESS IS GOOD
13212 064666 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13213 064672 104003              ERROR    +3            ;FPP ERROR
13214                                ;BAD RETURN ADDRESS
13215 064674 012600              5$:     MOV      (SP)+,R0     ;CLEAN UP STACK
13216 064676 012704 003314      MOV      #TAB6,R4      ;POINT TO EXPECTED DATA
13217 064702 004767 053172      JSR      R7,DATVER     ;VERIFY DATA
13218 064706 005767 116206      TST      COUNT
13219 064712 001403              BEQ      7$            ;BRANCH IF GOOD
13220 064714 004737 140132      CALL     @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
13221 064720 104003              ERROR    +3            ;FPP ERROR
13222                                ;BAD ACO
13223                                ;UNDERFLOW TRAPS DISABLED
13224                                ;
13225 064722 012702 000200      7$:     MOV      #200,R2     ;SETUP FLOATING POINT STATUS

```

FLOATING POINT TESTS

```

13226 064726 170102          LDFPS  R2          ;LOAD FPS
13227 064730 012737 140044 000244  MOV    @WLDTRP,@FPVEC ;REPLACE WILD TRAP VECTOR
13228 064736 012704 003334          MOV    @TAB7,R4      ;POINT TO FSRC DATA
13229 064742 012701 003714          MOV    @TAB41,R1     ;POINT TO ACO DATA
13230 064746 172411          LDD    (R1),ACO      ;LOAD ACO WITH TEST DATA
13231 064750 172014          ADDD   (R4),ACO      ;*TEST INSTRUCTION
13232 064752 170203          STFPS  R3           ;SAVE FPS
13233 064754 012701 003142  MOV    @RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13234 064760 174011          STD    ACO,(R1)     ;SAVE ACO RESULT
13235 064762 022703 000204  CMP    @204,R3       ;VERIFY STATUS
13236 064766 001403          BEQ    8$           ;BRANCH IF GOOD
13237 064770 004737 140132  CALL   @@DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13238 064774 104003          ERROR  +3          ;FPP ERROR
13239                                ;BAD FPS
13240 064776 012704 003314  8$:   MOV    @TAB6,R4      ;POINT TO EXPECTED DATA
13241 065002 004767 053072          JSR    R7,DATVER    ;VERIFY DATA
13242 065006 005767 116106          TST    COUNT
13243 065012 001401          BEQ    9$           ;BRANCH IF GOOD
13244 065014 104003          ERROR  +3          ;FPP ERROR
13245                                ;BAD ACO
13246                                ;UNDERFLOW TRAPS ENABLED
13247                                ;
13248 065016 012702 002200  9$:   MOV    @2200,R2     ;SETUP FLOATING POINT STATUS
13249 065022 170102          LDFPS  R2          ;LOAD FPS
13250 065024 012737 065056 000244  MOV    @11$,@FPVEC  ;REPOSITION TRAP VECTOR
13251 065032 012704 003334          MOV    @TAB7,R4      ;POINT TO FSRC DATA
13252 065036 012701 003714          MOV    @TAB41,R1     ;POINT TO ACO DATA
13253 065042 172411          LDD    (R1),ACO      ;LOAD ACO WITH TEST DATA
13254 065044 172014          ADDD   (R4),ACO      ;*TEST INSTRUCTION
13255 065046 170000          CFCC   ;COPY FPP CC
13256 065050 004737 140132  CALL   @@DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13257 065054 104003          ERROR  +3          ;FPP ERROR
13258                                ;FAILED TO TRAP ON UNDERFLOW
13259 065056 170203          11$:  STFPS  R3           ;SAVE FPS
13260 065060 012701 003142  MOV    @RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13261 065064 174011          STD    ACO,(R1)     ;SAVE ACO RESULT
13262 065066 022703 102210  CMP    @102210,R3    ;VERIFY STATUS
13263 065072 001403          BEQ    12$          ;BRANCH IF GOOD
13264 065074 004737 140132  CALL   @@DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13265 065100 104003          ERROR  +3          ;FPP ERROR
13266                                ;BAD FPS
13267 065102 012605          12$:  MOV    (SP)+,R5     ;GET ERROR PC
13268 065104 020527 065046  CMP    R5,@10$      ;VERIFY ERROR ADDRESS ON STACK
13269 065110 001403          BEQ    13$          ;BRANCH IF GOOD
13270 065112 004737 140132  CALL   @@DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13271 065116 104003          ERROR  +3          ;FPP ERROR
13272                                ;BAD ERROR RETURN ON STACK
13273 065120 005726          13$:  TST    (SP)+       ;RESTORE STACK
13274 065122 012704 003304  MOV    @TAB5A,R4     ;POINT TO EXPECTED DATA
13275 065126 004767 052746  JSR    R7,DATVER    ;VERIFY DATA
13276 065132 005767 115762          TST    COUNT
13277 065136 001403          BEQ    14$          ;BRANCH IF GOOD
13278 065140 004737 140132  CALL   @@DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13279 065144 104003          ERROR  +3          ;FPP ERROR
13280                                ;BAD ACO
13281                                ;UNDERFLOW WITH TRAPS DISABLED - NON ZERO RESULT
13282                                ;

```

FLOATING POINT TESTS

```

13283 065146 012702 000200      14$:  MOV      #200,R2          ;SETUP FLOATING POINT STATUS
13284 065152 170102              LDFPS    R2              ;LOAD FPS
13285 065154 012737 140044 000244  MOV      #WLDTRP,#FPVEC    ;RESTORE TRAP VECTOR
13286 065162 012704 003714      MOV      #TAB41,R4        ;POINT TO FSRC DATA
13287 065166 012701 003724      MOV      #TAB42,R1        ;POINT TO ACO DATA
13288 065172 172411              LDD      (R1),ACO        ;LOAD ACO WITH TEST DATA
13289 065174 172014              ADDD    (R4),ACO        ;*TEST INSTRUCTION
13290 065176 170203              ST+PS   R3              ;SAVE FPS
13291 065200 012701 003142      MOV      #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
13292 065204 174011              STD     ACO,(R1)        ;SAVE ACO RESULT
13293 065206 022703 000204      CMP     #204,R3         ;VERIFY STATUS
13294 065212 001403              BEQ     15$             ;BRANCH IF GOOD
13295 065214 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13296 065220 104003              ERROR   +3             ;FPP ERROR
13297                                ;BAD FPS
13298 065222 012704 003314      15$:  MOV      #TAB6,R4          ;POINT TO EXPECTED DATA
13299 065226 004767 052646      JSR     R7,DATVER      ;VERIFY DATA
13300 065232 005767 115662      TST     COUNT
13301 065236 001403              BEQ     16$             ;BRANCH IF GOOD
13302 065240 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13303 065244 104003              ERROR   +3             ;FPP ERROR
13304                                ;BAD ACO
13305                                ;JNERFLOW WITH TRAPS ENABLED
13306                                ;NON-ZERO RESULT
13307 065246 012702 102200      16$:  MOV      #102200,R2     ;SETUP FLOATING POINT STATUS
13308 065252 170102              LDFPS    R2              ;LOAD FPS
13309 065254 012737 065306 000244  MOV      #18$,#FPVEC    ;RESTORE TRAP VECTOR
13310 065262 012704 003714      MOV      #TAB41,R4        ;POINT TO FSRC DATA
13311 065266 012701 003724      MOV      #TAB42,R1        ;POINT TO ACO DATA
13312 065272 172411              LDD      (R1),ACO        ;LOAD ACO WITH TEST DATA
13313 065274 172014              ADDD    (R4),ACO        ;*TEST INSTRUCTION
13314 065276 170000              17$:  CFCC
13315 065300 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13316 065304 104003              ERROR   +3             ;FPP ERROR
13317                                ;NO TRAP ON UNDERFLOW
13318 065306 170203              18$:  STFPS   R3              ;SAVE FPS
13319 065310 012701 003142      MOV      #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
13320 065314 174011              STD     ACO,(R1)        ;SAVE ACO RESULT
13321 065316 012600              MOV      (SP)+,R0        ;SAVE STACK CONTENTS
13322 065320 005726              TST     (SP)+           ;CLEAN UP STACK
13323 065322 022700 065276      CMP     #17$,R0         ;VERIFY RETURN ADDRESS
13324 065326 00 403              BEQ     19$             ;BRANCH IF GOOD
13325 065330 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13326 065334 104003              ERROR   +3             ;FPP ERROR
13327                                ;BAD RETURN ADDRESS
13328 065336 022703 102204      19$:  CMP     #102204,R3     ;VERIFY STATUS
13329 065342 001403              BEQ     20$             ;BRANCH IF GOOD
13330 065344 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13331 065350 104003              ERROR   +3             ;FPP ERROR
13332                                ;BAD FPS
13333 065352 012704 003734      20$:  MOV      #TAB43,R4          ;POINT TO EXPECTED DATA
13334 065356 004767 052516      JSR     R7,DATVER      ;VERIFY DATA
13335 065362 005767 115532      TST     COUNT
13336 065366 001403              BEQ     21$             ;BRANCH IF GOOD
13337 065370 004737 140132      CALL    #DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13338 065374 104003              ERROR   +3             ;FPP ERROR
13339                                ;BAD ACO

```

FLOATING POINT TESTS

```

13340 065376      21$:
13341             ;
13344             ;
13345             ;-----
13346             ;TEST LDCFD, LDCDF
13347             ;
13348 065376      MLDC:
13349             ;
13350             ;TRUNCATE
13351 065376 012702 000300      MOV      #300,R2      ;SETUP FLOATING POINT STATUS
13352 065402 170102             LDFPS   R2           ;LOAD FPS
13353 065404 012704 003744      MOV      #TAB45,R4      ;POINT TO FSRC DATA
13354 065410 012701 003314      MOV      #TAB6,R1      ;POINT TO ACO DATA
13355 065414 172411             LDD      (R1),ACO      ;LOAD ACO WITH TEST DATA
13356 065416 177424             LDCFD   (R4)+,ACO      ;*TEST INSTRUCTION
13357 065420 012701 003142      MOV      #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13358 065424 174011             STD      ACO,(R1)      ;SAVE ACO RESULT
13359 065426 022704 003750      CMP      #TAB45+4,R4   ;VERIFY AUTO-INC
13360 065432 001403             BEQ      1$           ;BRANCH IF GOOD AUTO-INC
13361 065434 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
13362 065440 104003             ERROR   +3           ;FPP ERROR
13363             ;BAD AUTO-INC
13364 065442 012704 003754      1$:      MOV      #TAB46,R4      ;POINT TO EXPECTED DATA
13365 065446 004767 052426      JSR     R7,DATVER     ;VERIFY DATA
13366 065452 005767 115442      TST     COUNT
13367 065456 001403             BEQ      2$           ;BRANCH IF GOOD
13368 065460 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
13369 065464 104003             ERROR   +3           ;FPP ERROR
13370             ;BAD ACO
13371             ;AUTO-INC DOUBLE MODE
13372             ;
13373 065466 005002      2$:      CLR      R2           ;SETUP FLOATING POINT STATUS
13374 065470 170102             LDFPS   R2           ;LOAD FPS
13375 065472 012704 003744      MOV      #TAB45,R4      ;POINT TO FSRC DATA
13376 065476 012701 003464      MOV      #TAB16,R1     ;POINT TO ACO DATA
13377 065502 172411             LDD      (R1),ACO      ;LOAD ACO WITH TEST DATA
13378 065504 177424             LDCDF   (R4)+,ACO      ;*TEST INSTRUCTION
13379 065506 020427 003754      CMP      R4,#TAB45+10 ;VERIFY AUTO-INC
13380 065512 001403             BEQ      3$           ;BRANCH IF GOOD
13381 065514 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
13382 065520 104003             ERROR   +3           ;FPP ERROR
13383             ;BAD AUTO-INC ON DOUBLE
13384 065522 170203      3$:      STFPS   R3           ;SAVE FPS
13385 065524 012701 003142      MOV      #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13386 065530 174011             STD      ACO,(R1)      ;SAVE ACO RESULT
13387 065532 022703 000000      CMP      #0,R3        ;VERIFY STATUS
13388 065536 001403             BEQ      4$           ;BRANCH IF GOOD
13389 065540 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
13390 065544 104003             ERROR   +3           ;FPP ERROR
13391             ;BAD FPS
13392 065546 012704 004014      4$:      MOV      #TAB49,R4      ;POINT TO EXPECTED DATA
13393 065552 004767 052322      JSR     R7,DATVER     ;VERIFY DATA
13394 065556 005767 115336      TST     COUNT
13395 065562 001403             BEQ      5$           ;BRANCH IF GOOD
13396 065564 004737 140132      CALL    @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
13397 065570 104003             ERROR   +3           ;FPP ERROR
13398             ;BAD ACO

```


FLOATING POINT TESTS

```

13399          ;LDCFD GR7
13400          ;
13401 065572 012702 000200 5$:  MOV    #200,R2          ;SETUP FLOATING POINT STATUS
13402 065576 170102          LDFPS  R2          ;LOAD FPS
13403 065600 005003          CLR    R3
13404 065602 177427 043243  LDCFD  #5203,ACO      ;*TEST INSTRUCTION
13405 065606 005203          INC    R3
13406 065610 005203          INC    R3
13407 065612 005203          INC    R3          ;IF LDCFD WORKED, R3 SHOULD=?
13408 065614 022703 000003  CMP    #3,R3        ;VERIFY CORRECT PROGRAM FLOW
13409 065620 001403          BEQ    6$          ;BRANCH IF GOOD
13410 065622 004737 140132  CALL   @#DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13411 065626 104003          ERROR  +3          ;FPP ERROR
13412          ;BAD PROGRAM FLOW
13413          ;
13414          ;NEGATIVE OPERANDS
13415 065630 012702 000200 6$:  MOV    #200,R2          ;SETUP FLOATING POINT STATUS
13416 065634 170102          LDFPS  R2          ;LOAD FPS
13417 065636 012704 003764  MOV    #TAB47,R4     ;POINT TO FSRC DATA
13418 065642 012701 003744  MOV    #TAB45,R1     ;POINT TO ACO DATA
13419 065646 172411          LDD    (R1),ACO      ;LOAD ACO WITH TEST DATA
13420 065650 177414          LDCFD  (R4),ACO      ;*TEST INSTRUCTION
13421 065652 170203          STFPS  R3          ;SAVE FPS
13422 065654 012701 003142  MOV    #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13423 065660 174011          STD    ACO,(R1)     ;SAVE ACO RESULT
13424 065662 022703 000210  CMP    #210,R3      ;VERIFY STATUS
13425 065666 001403          BEQ    7$          ;BRANCH IF GOOD
13426 065670 004737 140132  CALL   @#DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13427 065674 104003          ERROR  +3          ;FPP ERROR
13428          ;BAD FPS
13429 065676 012704 004004 7$:  MOV    #TAB48,R4     ;POINT TO EXPECTED DATA
13430 065702 004767 052172  JSR    R7,DATVER     ;VERIFY DATA
13431 065706 005767 115206  TST    COUNT
13432 065712 001403          BEQ    8$          ;BRANCH IF GOOD
13433 065714 004737 140132  CALL   @#DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13434 065720 104003          ERROR  +3          ;FPP ERROR
13435          ;BAD ACO
13436          ;LOAD A ZERO
13437          ;
13438 065722 012702 000200 8$:  MOV    #200,R2          ;SETUP FLOATING POINT STATUS
13439 065726 170102          LDFPS  R2          ;LOAD FPS
13440 065730 012704 0C3314  MOV    #TAB6,R4      ;POINT TO FSRC DATA
13441 065734 012701 004004  MOV    #TAB48,R1     ;POINT TO ACO DATA
13442 065740 172411          LDD    (R1),ACO      ;LOAD ACO WITH TEST DATA
13443 065742 177414          LDCFD  (R4),ACO      ;*TEST INSTRUCTION
13444 065744 170203          STFPS  R3          ;SAVE FPS
13445 065746 012701 003142  MOV    #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
13446 065752 174011          STD    ACO,(R1)     ;SAVE ACO RESULT
13447 065754 022703 000204  CMP    #204,R3      ;VERIFY STATUS
13448 065760 001403          BEQ    9$          ;BRANCH IF GOOD
13449 065762 004737 140132  CALL   @#DETFPA     ;DETERMINE FLOATING POINT FAULT. $$$
13450 065766 104003          ERROR  +3          ;FPP ERROR
13451          ;BAD FPS
13452 065770 012704 003314 9$:  MOV    #TAB6,R4      ;POINT TO EXPECTED DATA
13453 065774 004767 052100  JSR    R7,DATVER     ;VERIFY DATA
13454 066000 005767 115114  TST    COUNT
13455 066004 001403          BEQ    10$         ;BRANCH IF GOOD

```


FLOATING POINT TESTS

```

13507      ; (FEC)
13508      ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13509      ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13510      ;
13511 066176 012605 CMPRTN: MOV (SP)+,R5 ; RETURN ADDRESS TO USE AS POINTER
13512 066200 012702 000200      MOV #200,R2 ; SET TO DOUBLE MODE FOR LOAD
13513 066204 170102      LDFPS R2 ; LOAD FPS
13514 066206 010504      MOV R5,R4 ; POINT TO FSRC DATA
13515 066210 062704 000010      ADD #10,R4 ;
13516 066214 010501      MOV R5,R1 ; POINT TO ACO DATA
13517 066216 172411      LDD (R1),ACO ; LOAD ACO WITH TEST DATA
13518 066220 016502 000020      MOV 20(R5),R2 ; GET TEST FPS
13519 066224 170102      LDFPS R2 ; LOAD TEST FPS
13520 066226 173414 1$: CMPD (R4),ACO ; *TEST INSTRUCTION
13521 066230 170203      STFPS R3 ; SAVE FPS
13522 066232 012702 000200      MOV #200,R2 ; SET FPP TO DOUBLE
13523 066236 170102      LDFPS R2 ;
13524 066240 012701 003142      MOV #RECDST,R1 ; POINT TO RECEIVED DATA TABLE
13525 066244 174011      STD ACO,(R1) ; SAVE ACO RESULT
13526 066246 026503 000022      CMP 22(R5),R3 ; VERIFY STATUS
13527 066252 001403      BEQ 2$ ; BRANCH IF GOOD
13528 066254 004737 140132      CALL #DETFPA ; DETERMINE FLOATING POINT FAULT. $$$
13529 066260 104003      ERROR +3 ; FPP ERROR
13530      ; BAD FPS
13531 066262 005737 003030 2$: TST #FLAG ; SEE IF ACO REMAINS UNCHANGED
13532 066266 001403      BEQ 3$ ; BRANCH IF ACO STAYS THE SAME
13533 066270 012704 003314      MOV #TAB6,R4 ; ACO=0
13534 066274 000401      BR 4$ ; GO VERIFY DATA
13535 066276 010504 3$: MOV R5,R4 ; POINT TO EXPECTED DATA
13536 066300 004767 051574 4$: JSR R7,DATVER ; VERIFY DATA
13537 066304 005767 114610      TST COUNT ;
13538 066310 001403      BEQ 5$ ; BRANCH IF GOOD
13539 066312 004737 140132      CALL #DETFPA ; DETERMINE FLOATING POINT FAULT. $$$
13540 066316 104003      ERROR +3 ; FPP ERROR
13541      ; BAD ACO
13542 066320 000165 000024 5$: JMP 24(R5) ; RETURN
13543 066324      HOP44:
13544      ;
13547      ;
13548      ; -----
13549      ; TEST DIVF
13550      ;
13551 066324      MDIVF:
13552      ;
13553      ; 1/EXP[AC]=FSRC=0
13554 066324 012737 000002 003030      MOV #2,#FLAG ; NO INTERRUPT, BUT FEC
13555 066332 004767 000706      JSR R7,DVFSUB ; DO TEST
13556 066336 000100 000027      .WORD 100,27 ; ACO
13557 066342 00000C 000000      .WORD 0,0 ; FSRC
13558 066346 000100 000027      .WORD 100,27 ; RESULT
13559 066352 040000      .WORD 40000 ; TEST FPS
13560 066354 140000      .WORD 140000 ; RESULT FPS
13561 066356 000004      .WORD 4 ; FEC
13562      ; 2/AC=EXP[FSRC]=0
13563      ; TRAPS ENABLED
13564 066360 012737 000001 003030      MOV #1,#FLAG ; INTERRUPT
13565 066366 004767 000652      JSR R7,DVFSUB ; DO TEST

```

FLOATING POINT TESTS

13566	066372	000000	000000		.WORD	0,0		;ACO
13567	066376	000100	000000		.WORD	100,0		;FSRC
13568	066402	000000	000000		.WORD	0,0		;RESULT
13569	066406	000000			.WORD	0		; TEST FPS
13570	066410	100000			.WORD	100000		;RESULT FPS
13571	066412	000004			.WORD	4		;FEC
13572								
13573	066414	005037	003030					
13574	066420	004767	000620		CLR	0#FLAG		;NO INTERRUPT
13575	066424	000177	000234		JSR	R7,DVFSUB		;DO TEST
13576	066430	004100	000000		.WORD	177,234		;ACO
13577	066434	000000	000000		.WORD	4100,0		;FSRC
13578	066440	007400			.WORD	0,0		;RESULT
13579	066442	007404			.WORD	7400		; TEST FPS
13580					.WORD	7404		;RESULT FPS
13581	066444	012737	000001	003030				
13582	066452	004767	000056					
13583	066456	040200	104210					
13584	066462	000125	025252					
13585	066466	040200	104210					
13586	066472	007557						
13587	066474	107557						
13588	066476	000004						
13589								
13590	066500	005037	003030					
13591	066504	004767	000534					
13592	066510	077760	177777					
13593	066514	077760	000000					
13594	066520	040200	104210					
13595	066524	007414						
13596	066526	007400						
13597								
13598	066530	005037	003030					
13599	066534	004767	000504					
13600	066540	052525	052525					
13601	066544	052525	052525					
13602	066550	040200	000000					
13603	066554	007400						
13604	066556	007400						
13605								
13606	066560	005037	003030					
13607	066564	004767	000454					
13608	066570	077777	125252					
13609	066574	040300	000000					
13610	066600	077652	070707					
13611	066604	007400						
13612	066606	007400						
13613								
13614	066610	005037	003030					
13615	066614	004767	000424					
13616	066620	055377	177777					
13617	066624	055300	000000					
13618	066630	040252	125252					
13619	066634	000000						
13620	066636	000000						
13621								
13622	066640	005037	003030					

FLOATING POINT TESTS

```

13623 066644 004767 000374      JSR      R7,DVFSUB          ;DO TEST
13624 056650 064600 000001      .WORD   64600,1           ;ACO
13625 066654 066600 000000      .WORD   66600,0           ;FSRC
13626 066660 036200 000001      .WORD   36200,1           ;RESULT
13627 066664 000000      .WORD   0                 ; TEST FPS
13628 066666 000000      .WORD   0                 ;RESULT FPS
13629      ;10/AC>FSRC>0
13630 066670 005037 003030      CLR      @#FLAG           ;NO INTERRUPT
13631 066674 004767 000344      JSR      R7,DVFSUB          ;DO TEST
13632 066700 012345 156024      .WORD   12345,156024      ;ACO
13633 066704 005600 000000      .WORD   05600,0           ;FSRC
13634 066710 044745 156024      .WORD   44745,156024      ;RESULT
13635 066714 000017      .WORD   17                ; TEST FPS
13636 066716 000000      .WORD   0                 ;RESULT FPS
13637      ;11/FSRC<0
13638 066720 005037 003030      CLR      @#FLAG           ;NO INTERRUPT
13639 066724 004767 000314      JSR      R7,DVFSUB          ;DO TEST
13640 066730 040422 101010      .WORD   40422,101010      ;ACO
13641 066734 140511 101010      .WORD   140511,101010     ;FSRC
13642 066740 140072 020167      .WORD   140072,20167      ;RESULT
13643 066744 000057      .WORD   57                ; TEST FPS
13644 066746 000050      .WORD   50                ;RESULT FPS
13645      ;12/AC<0
13646 066750 005037 003030      CLR      @#FLAG           ;NO INTERRUPT
13647 066754 004767 000264      JSR      R7,DVFSUB          ;DO TEST
13648 066760 160077 000101      .WORD   160077,101        ;ACO
13649 066764 040417 177777      .WORD   40417,-1         ;FSRC
13650 066770 157651 143527      .WORD   157651,143527     ;RESULT
13651 066774 000007      .WORD   7                 ; TEST FPS
13652 066776 000010      .WORD   10                ;RESULT FPS
13653      ;13/TRUNCATE TEST
13654 067000 005037 003030      CLR      @#FLAG           ;NO INTERRUPT
13655 067004 004767 000234      JSR      R7,DVFSUB          ;DO TEST
13656 067010 060100 000177      .WORD   60100,177        ;ACO
13657 067014 040300 000000      .WORD   40300,0           ;FSRC
13658 067020 060000 000124      .WORD   60000,124        ;RESULT
13659 067024 000040      .WORD   40                ; TEST FPS
13660 067026 000040      .WORD   40                ;RESULT FPS
13661      ;14/ROUND TEST
13662 067030 005037 003030      CLR      @#FLAG           ;NO INTERRUPT
13663 067034 004767 000204      JSR      R7,DVFSUB          ;DO TEST
13664 067040 060100 000177      .WORD   60100,177        ;ACO
13665 067044 040300 000000      .WORD   40300,0 ;FSRC
13666 067050 060000 000125      .WORD   60000,125        ;RESULT
13667 067054 000000      .WORD   0                 ; TEST FPS
13668 067056 000000      .WORD   0                 ;RESULT FPS
13669      ;15/OVERFLOW, INTERRUPTS ENABLED
13670 067060 012737 000001 003030      MOV      #1,@#FLAG        ;INTERRUPT
13671 067066 004767 000152      JSR      R7,DVFSUB          ;DO TEST
13672 067072 177700 000000      .WORD   177700,0         ;ACO
13673 067076 000200 000000      .WORD   200,0             ;FSRC
13674 067102 137700 000000      .WORD   137700,0         ;RESULT
13675 067106 001100      .WORD   1100              ; TEST FPS
13676 067110 101112      .WORD   101112           ;RESULT FPS
13677 067112 000010      .WORD   10                ;FEC
13678      ;16/OVERFLOW, TRAPS DISABLED
13679 067114 012737 000002 003030      MOV      #2,@#FLAG        ;NO INTERRUPT

```

FLOATING POINT TESTS

```

13680 067122 004767 000116      JSR      R7,DVFSUB      ;DO TEST
13681 067126 000200 000000      .WORD   200,0          ;ACO
13682 067132 177700 000000      .WORD   177700,0      ;FSRC
13683 067136 000000 000000      .WORD   0,0           ;RESULT
13684 067142 041100      .WORD   41100         ; TEST FPS
13685 067144 041104      .WORD   41104         ;RESULT FPS
13686 067146 000010      .WORD   10            ;FEC OVERFLOW
13687                                     ;17/UNDERFLOW, TRAPS ENABLED, UV RESULT
13688 067150 012737 000001 003030      MOV     #1,0#FLAG      ;INTERRUPT
13689 067156 004767 000062      JSR     R7,DVFSUB      ;DO TEST
13690 067162 100200 000000      .WORD   100200,0      ;ACO
13691 067166 040377 177777      .WORD   40377,-1     ;FSRC
13692 067172 100000 000001      .WORD   100000,1     ;RESULT
13693 067176 002000      .WORD   2000          ; TEST FPS
13694 067200 102014      .WORD   102014        ;RESULT FPS
13695 067202 000012      .WORD   12            ;FEC
13696                                     ;18/UNDERFLOW, TRAPS ENABLED, ROUND
13697 067204 012737 000001 003030      MOV     #1,0#FLAG      ;INTERRUPT
13698 067212 004767 000026      JSR     R7,DVFSUB      ;DO TEST
13699 067216 030325 025252      .WORD   30325,25252   ;ACO
13700 067222 076777 023456      .WORD   76777,23456   ;FSRC
13701 067226 071525 157716      .WORD   71525,157716 ;RESULT
13702 067232 002537      .WORD   2537          ; TEST FPS
13703 067234 102500      .WORD   102500        ;RESULT FPS
13704 067236 000012      .WORD   12            ;FEC
13705                                     ;
13706 067240 000167 000242      JMP     HOP10          ;GO TO NEXT TEST
13707                                     ;*****
13708                                     ;*****
13709                                     ;DIVF SUBROUTINE:
13710                                     ;
13711                                     ;       ACO
13712                                     ;       FSRC
13713                                     ;       FPS BEFORE EXECUTION
13714                                     ;       FPS AFTER EXECUTION
13715                                     ;       (FEC)
13716                                     ;*****
13717                                     ;*****
13718                                     ;
13719 067244 012605      DVFSUB: MOV     (SP)+,R5      ; RETURN ADDRESS TO USE AS POINTER
13720 067246 012737 067326 000244      MOV     #50#,0#FPVEC   ;REDIRECT TRAP VECTOR
13721 067254 012702 000200      MOV     #200,R2        ;SET TO DOUBLE MODE FOR LOAD
13722 067260 170102      LDFPS  R2              ;LOAD FPS
13723 067262 010504      MOV     R5,R4          ;POINT TO FSRC DATA
13724 067264 062704 000004      ADD     #4,R4
13725 067270 172415      LDD     (R5),ACO       ;LOAD ACO WITH TEST DATA
13726 067272 016502 000014      MOV     14(R5),R2      ;GET TEST FPS
13727 067276 170102      LDFPS  R2              ;LOAD TEST FPS
13728                                     ;
13729 067300 174414      DIVF   (R4),ACO        ;*TEST INSTRUCTION
13730 067302 170001      1$:   SETF              ;WAIT FOR POSSIBLE FPA TRAP.
13731                                     ;
13732                                     ;INSTRUCTION DIDNT TRAP
13733                                     ;
13734 067304 032737 000001 003030      BIT     #1,0#FLAG      ;VERIFY A NO TRAP CONDITION
13735 067312 001426      BEQ    2$              ;BRANCH IF GOOD
13736 067314 004737 140132      CALL   0#DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$

```


FLOATING POINT TESTS

```

13796
13797
13798 067506 012757 000002 003030
13799 067514 004767 000516
13800 067520 000000 000000 000000
      067526 000001
13801 067530 000100 000000 000000
      067536 000000
13802 067540 000000 000000 000000
      067546 000001
13803 067550 040000
13804 067552 140000
13805 067554 000004
13806
13807 067556 012737 000001 003030
13808 067564 004767 000446
13809 067570 000402 000000 000000
      067576 000000
13810 067600 000000 000000 000000
      067606 000000
13811 067610 000402 000000 000000
      067616 000000
13812 067620 000200
13813 067622 100200
13814 067624 000004
13815
13816 067626 005037 003030
13817 067632 004767 000400
13818 067636 034300 000000 000000
      067644 000001
13819 067646 140300 000000 000000
      067654 000000
13820 067656 134200 000000 000000
      067664 000001
13821 067666 000200
13822 067670 000210
13823
13824 067672 005037 003030
13825 067676 004767 000334
13826 067702 034300 000000 000000
      067710 000001
13827 067712 140300 000000 000000
      067720 000000
13828 067722 134200 000000 000000
      067730 000000
13829 067732 000240
13830 067734 000250
13831
13832 067736 005037 003030
13833 067742 004767 000270
13834 067746 177642 000000 000000
      067754 000151
13835 067756 166600 000000 000000
      067764 000123
13836 067766 051242 000000 000000
      067774 000000
13837 067776 000200

```

;1/AC=FSRC=0 TRAPS DISABLED
MOV #2,#FLAG ;NO INTERRUPT
JSR R7,DVDSUB ;DO TEST
.WORD 0,0,0,1 ;ACO
.WORD 100,0,0,0 ;FSRC
.WORD 0,0,0,1 ;RESULT
.WORD 40000 ; TEST FPS
.WORD 140000 ;RESULT FPS
.WORD 4 ;FEC
;2/FSRC=0, TRAPS ENABLED
MOV #1,#FLAG ;INTERRUPT
JSR R7,DVDSUB ;DO TEST
.WORD 402,0,0,0 ;ACO
.WORD 0,0,0,0 ;FSRC
.WORD 402,0,0,0 ;RESULT
.WORD 200 ; TEST FPS
.WORD 100200 ;RESULT FPS
.WORD 4 ;FEC
;3/ROUND
CLR #FLAG ;NO INTERRUPT
JSR R7,DVDSUB ;DO TEST
.WORD 34300,0,0,1 ;ACO
.WORD 140300,0,0,0 ;FSRC
.WORD 134200,0,0,1 ;RESULT
.WORD 200 ; TEST FPS
.WORD 210 ;RESULT FPS
;4/TRUNCATE
CLR #FLAG ;NO INTERRUPT
JSR R7,DVDSUB ;DO TEST
.WORD 34300,0,0,1 ;ACO
.WORD 140300,0,0,0 ;FSRC
.WORD 134200,0,0,0 ;RESULT
.WORD 240 ; TEST FPS
.WORD 250 ;RESULT FPS
;5/ROUND NEGATIVE AC, FSRC
CLR #FLAG ;NO INTERRUPT
JSR R7,DVDSUB ;DO TEST
.WORD 177642,0,0,151 ;ACO
.WORD 166600,0,0,123 ;FSRC
.WORD 51242,0,0,0 ;RESULT
.WORD 200 ; TEST FPS

FLOATING POINT TESTS

```

13838 070000 000200      .WORD 200      ;RESULT FPS
13839                    ;6/TRUNCATE NEAGTIVE AC, FSRC
13840 070002 005037 003030  CLR      @FLAG ;NO INTERRUPT
13841 070006 004767 000224  JSR      R7,DVDSUB ;DO TEST
13842 070012 177642 000000 000000  .WORD 177642,0,0,151 ;ACO
      070020 000151
13843 070022 166600 000000 000000  .WORD 166600,0,0,123 ;FSRC
      070030 000123
13844 070032 051241 177777 177777  .WORD 51241,-1,-1,1 ;RESULT
      070040 177777
13845 070042 000240      .WORD 240      ; TEST FPS
13846 070044 000240      .WORD 240      ;RESULT FPS
13847                    ;7/AC=FSRC
13848 070046 005037 003030  CLR      @FLAG ;NO INTERRUPT
13849 070052 004767 000160  JSR      R7,DVDSUB ;DO TEST
13850 070056 055521 047621 100333  .WORD 55521,47621,100333,-1 ;ACO
      070064 177777
13851 070066 055521 047621 100333  .WORD 55521,47621,100333,-1 ;FSRC
      070074 177777
13852 070076 040200 000000 000000  .WORD 40200,0,0,0 ;RESULT
      070104 000000
13853 070106 007717      .WORD 7717      ; TEST FPS
13854 070110 007700      .WORD 7700      ;RESULT FPS
13855                    ;8/UNDERFLOW TRAPS ENABLED, UV RESULT
13856 070112 012737 000001 003030  MOV      #1,@FLAG ;INTERRUPT
13857 070120 004767 000112  JSR      R7,DVDSUB ;DO TEST
13858 070124 100200 000000 000000  .WORD 100200,0,0,0 ;ACO
      070132 000000
13859 070134 077777 000000 000000  .WORD 77777,0,0,0 ;FSRC
      070142 000000
13860 070144 140400 100200 100200  .WORD 140400,100200,100200,100201 ;RESULT
      070152 100201
13861 070154 002200      .WORD 2200      ; TEST FPS
13862 070156 102210      .WORD 102210     ;RESULT FPS
13863 070160 000012      .WORD 12         ;FEC
13864                    ;9/OVERFLOW TRAPS ENABLED
13865 070162 012737 000001 003030  MOV      #1,@FLAG ;INTERRUPT
13866 070170 004767 000042  JSR      R7,DVDSUB ;DO TEST
13867 070174 077000 123465 012346  .WORD 77000,123465,12346.525 ;ACO
      070202 000525
13868 070204 000303 000001 140000  .WORD 303,1,140000,140001 ;FSRC
      070212 140001
13869 070214 036650 163002 103645  .WORD 36650,163002,103645,64003 ;RESULT
      070222 064003
13870 070224 001700      .WORD 1700      ; TEST FPS
13871 070226 101702      .WORD 101702     ;RESULT FPS
13872 070230 000010      .WORD 10         ;FEC
13873                    ;
13874 070232 000167 000242  JMP      HOP11 ;HOP OVER SUBROUTINE
13875                    ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
13876                    ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
13877                    ;
13878                    ;DIVD SUBROUTINE:
13879                    ; ACO
13880                    ; FSRC
13881                    ; FPS BEFCRE EXECUTION
13882                    ; FPS AFTER EXECUTION

```

FLOATING POINT TESTS

```

13883             ;             (FEC)
13884             ;
13885             ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13886             ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13887             ;
13888 070236 012605 DVDSUB: MOV      (SP)+,R5           ; RETURN ADDRESS TO USE AS POINTER
13889 070240 012737 070320 000244     MOV      #50$,@#FPVEC       ; REDIRECT TRAP VECTOR
13890 070246 012702 000200           MOV      #200,R2          ; SET TO DOUBLE MODE FOR LOAD
13891 070252 170102           LDFPS   R2                ; LOAD FPS
13892 070254 010504           MOV      R5,R4            ; POINT TO FSRC DATA
13893 070256 062704 000010           ADD      #10,R4
13894 070262 172415           LD      (R5),ACO         ; LOAD ACO WITH TEST DATA
13895 070264 016502 000030           MOV      30(R5),R2       ; GET TEST FPS
13896 070270 170102           LDFPS   R2                ; LOAD TEST FPS
13897             ;
13898 070272 174414           DIVD    (R4),ACO         ; *TEST INSTRUCTION
13899 070274 170000 1$:          CFCC                   ; WAIT FOR POSSIBLE FPA TRAP.
13900             ;
13901             ; INSTRUCTION DIDNT TRAP
13902             ;
13903 070276 032737 000001 003030     BIT      #1,@#FLAG       ; VERIFY A NO TRAP CONDITION
13904 070304 001426           BEQ     2$               ; BRANCH IF GOOD
13905 070306 004737 140132           CALL    @#DETFPA         ; DETERMINE FLOATING POINT FAULT. $$$
13906 070312 104003           ERROR  +3                ; FPP ERROR
13907             ;
13908 070314 000167 000042           JMP     2$               ; INSTRUCTION SHOULD HAVE TRAPPED
13909             ;
13910             ; INSTRUCTION TRAPPED
13911             ;
13912 070320 032737 000001 003030 50$:   BIT      #1,@#FLAG       ; SEE IF EXPECTING A TRAP
13913 070326 001005           BNE    51$              ; BRANCH IF EXPECTING A TRAP
13914 070330 004737 140132           CALL    @#DETFPA         ; DETERMINE FLOATING POINT FAULT. $$$
13915 070334 104003           ERROR  +3                ; FPP ERROR
13916             ;
13917 070336 000167 000020           JMP     2$               ; INSTRUCTION WASNT SUPPOSE TO TRAP
13918 070342 012604 51$:          MOV      (SP)+,R4         ; REJOIN CODE
13919 070344 005726           TST    (SP)+             ; SEE IF PC = INSTRUCTION
13920 070346 022704 070274           CMP     #1$,R4           ; CLEAN UP STACK
13921 070352 001403           BEQ     2$               ;
13922 070354 004737 140132           CALL    @#DETFPA         ; BRANCH IF GOOD COMPARE
13923 070360 104003           ERROR  +3                ; DETERMINE FLOATING POINT FAULT. $$$
13924             ;
13925             ;
13926             ; COMMON CODE FOR TRAP AND NO TRAP
13927             ;
13928 070362 170203 2$:          STFPS   R3                ; SAVE FPS
13929 070364 012702 000200           MOV      #200,R2          ; SET FPP TO DOUBLE
13930 070370 170102           LDFPS   R2
13931 070372 012701 003142           MOV      @#RECDST,R1     ; POINT TO RECEIVED DATA TABLE
13932 070376 174011           STD     ACO,(R1)         ; SAVE ACO RESULT
13933 070400 026503 000032           CMP     32(R5),R3        ; VERIFY STATUS
13934 070404 001403           BEQ     3$               ; BRANCH IF GOOD
13935 070406 004737 140132           CALL    @#DETFPA         ; DETERMINE FLOATING POINT FAULT. $$$
13936 070412 104003           ERROR  +3                ; FPP ERROR
13937             ;
13938 070414 010504 3$:          MOV      R5,R4            ; BAD FPS
13939 070416 062704 000020           ADD     #20,R4           ; POINT TO EXPECTED DATA

```

FLOATING POINT TESTS

```

13940 070422 004767 047452      4$:   JSR   R7,DATVER           ;VERIFY DATA
13941 070426 005767 112466      TST   COUNT
13942 070432 001403              BEQ   5$                     ;BRANCH IF GOOD
13943 070434 004737 140132      CALL  @DETFPA             ;DETERMINE FLOATING POINT FAULT. $$$
13944 070440 104003              ERROR  +3                   ;FPP ERROR
13945                                ;BAD ACO
13946 070442 005737 003030      5$:   TST   @FLAG           ;SEE IF NEED TO CHECK FEC
13947 070446 001002              BNE   6$                     ;BRANCH IF NEED TO CHECK
13948 070450 000165 000034      JMP   34(R5)                ;RETURN FROM TEST
13949 070454 170301              6$:   STST  R1                 ;SAVE FEC
13950 070456 016504 000034      MOV   34(R5),R4            ;GET FEC
13951 070462 020401              CMP   R4,R1                 ;VERIFY FEC
13952 070464 001403              BEQ   7$                     ;BRANCH IF GOOD
13953 070466 004737 140132      CALL  @DETFPA             ;DETERMINE FLOATING POINT FAULT. $$$
13954 070472 104003              ERROR  +3                   ;FPP ERROR
13955                                ;BAD FEC
13956 070474 000165 000036      7$:   JMP   36(R5)            ;RETURN FROM TEST
13957
13958 070500      HOP11:
13961
13962
13963      ;-----
13964      ;TEST MULF
13965 070500      MMULF:
13966
13967      ;1/AC=FSRC=0 - INTERRUPTS DISABLED
13968 070500 005037 003030      CLR   @FLAG                 ;NO INTERRUPT
13969 070504 004767 000564      JSR   R7,MLFSUB             ;DO TEST
13970 070510 000000 000000      .WORD 0,0                   ;AC0
13971 070514 000000 000000      .WORD 0,0                   ;FSRC
13972 070520 000000 000000      .WORD 0,0                   ;RESULT
13973 070524 007517              .WORD 7517                   ; TEST FPS
13974 070526 007504              .WORD 7504                   ;RESULTANT FPS
13975
13976 070530 005037 003030      ;2/AC>FSRC=0 - INTERRUPTS ON
13977 070534 004767 000534      CLR   @FLAG                 ;NO INTERRUPT
13978 070540 000200 000000      JSR   R7,MLFSUB             ;DO TEST
13979 070544 000000 000000      .WORD 200,0                 ;AC0
13980 070550 000000 000000      .WORD 0,0                   ;FSRC
13981 070554 000013              .WORD 13                     ;RESULT
13982 070556 000004              .WORD 4                      ; TEST FPS
13983                                ;RESULTANT FPS
13984 070560 005037 003030      ;3/AC=0 FSRC>0 -
13985 070564 004767 000504      CLR   @FLAG                 ;NO INTERRUPT
13986 070570 000100 000000      JSR   R7,MLFSUB             ;DO TEST
13987 070574 000300 000000      .WORD 100,0                 ;AC0
13988 070600 000000 000000      .WORD 300,0                 ;FSRC
13989 070604 007500              .WORD 0,0                   ;RESULT
13990 070606 007504              .WORD 7500                   ; TEST FPS
13991                                ;RESULTANT FPS
13992 070610 005037 003030      ;4/AC=1 >FSRC - ROUND
13993 070614 004767 000454      CLR   @FLAG                 ;NO INTERRUPT
13994 070620 040200 000000      JSR   R7,MLFSUB             ;DO TEST
13995 070624 040177 177777      .WORD 40200,0               ;AC0
13996 070630 040177 177777      .WORD 40177,-1              ;FSRC
13997 070634 000000              .WORD 40177,-1              ;RESULT
13998 070636 000000              .WORD 0                      ; TEST FPS
13999                                ;RESULTANT FPS

```

FLOATING POINT TESTS

```

13999
14000 070640 005037 003030      ;5/TRUNCATE
14001 070644 004767 000424      CLR      @FLAG      ;NO INTERRUPT
14002 070650 040177 177777      JSR      R7,MLFSUB  ;DO TEST
14003 070654 040200 000000      .WORD   40177,-1    ;ACO
14004 070660 040177 177777      .WORD   40200,0 ;FSRC
14005 070664 000040      .WORD   40177,-1    ;RESULT
14006 070666 000040      .WORD   40          ; TEST FPS
14007      .WORD   40          ;RESULTANT FPS
14008 070670 005037 003030      ;6/NORMALIZE
14009 070674 004767 000374      CLR      @FLAG      ;NO INTERRUPT
14010 070700 040100 000000      JSR      R7,MLFSUB  ;DO TEST
14011 070704 040100 000000      .WORD   40100,0 ;ACO
14012 070710 040020 000000      .WORD   40100,0 ;FSRC
14013 070714 000012      .WORD   40020,0    ;RESULT
14014 070716 000000      .WORD   12          ; TEST FPS
14015      .WORD   0          ;RESULTANT FPS
14016 070720 005037 003030      ;7/ROUND
14017 070724 004767 000344      CLR      @FLAG      ;NO INTERRUPT
14018 070730 017500 000000      JSR      R7,MLFSUB  ;DO TEST
14019 070734 023652 125252      .WORD   17500,0 ;ACO
14020 070740 003177 177777      .WORD   23652,125252 ;FSRC
14021 070744 007417      .WORD   3177,-1    ;RESULT
14022 070746 007400      .WORD   7417        ; TEST FPS
14023      .WORD   7400        ;RESULTANT FPS
14024 070750 005037 003030      ;8/AC>0>FSRC ROUND
14025 070754 004767 000314      CLR      @FLAG      ;NO INTERRUPT
14026 070760 040342 177777      JSR      R7,MLFSUB  ;DO TEST
14027 070764 176543 025252      .WORD   40342,-1    ;ACO
14028 070770 176711 067324      .WORD   176543,025252 ;FSRC
14029 070774 007500      .WORD   176711,67324 ;RESULT
14030 070776 007510      .WORD   7500        ; TEST FPS
14031      .WORD   7510        ;RESULTANT FPS
14032 071000 005037 003030      ;9/IAC<FSRC<0, ROUND
14033 071004 004767 000264      CLR      @FLAG      ;NO INTERRUPT
14034 071010 144600 000000      JSR      R7,MLFSUB  ;DO TEST
14035 071014 154000 000000      .WORD   144600,0    ;ACO
14036 071020 060400 000000      .WORD   154000,0    ;FSRC
14037 071024 000017      .WORD   60400,0    ;RESULT
14038 071026 000000      .WORD   17          ; TEST FPS
14039      .WORD   0          ;RESULT FPS
14040 071030 005037 003030      ;10/AC<FSRC, ROUND
14041 071034 004767 000234      CLR      @FLAG      ;NO INTERRUPT
14042 071040 060000 000000      JSR      R7,MLFSUB  ;DO TEST
14043 071044 140377 177776      .WORD   60000,0     ;ACO
14044 071050 160177 177776      .WORD   140377,177776 ;FSRC
14045 071054 000017      .WORD   160177,177776 ;RESULT
14046 071056 000010      .WORD   17          ; TEST FPS
14047      .WORD   10          ;RESULT FPS
14048 071060 005037 003030      ;11/AC>0>FSRC, TRUNCATE
14049 071064 004767 000204      CLR      @FLAG      ;NO INTERRUPT
14050 071070 060000 000000      JSR      R7,MLFSUB  ;DO TEST
14051 071074 140377 177776      .WORD   60000,0     ;ACO
14052 071100 160177 177776      .WORD   140377,177776 ;FSRC
14053 071104 007547      .WORD   160177,177776 ;RESULT
14054 071106 007550      .WORD   7547        ; TEST FPS
14055      .WORD   7550        ;RESULT FPS
;12/UNDERFLOW, NO INTERRUPTS

```

FLOATING POINT TESTS

```

14056 071110 012737 000002 003030      MOV      #2,#FLAG          ;NO INTERRUPT
14057 071116 004767 000152              JSR      R7,MLFSUB        ;DO TEST
14058 071122 000200 000001              .WORD   200.1            ;ACO
14059 071126 000200 000001              .WORD   200.1            ;FSRC
14060 071132 040200 000002              .WORD   40200.2         ;RESULT
14061 071136 042117                    .WORD   42117           ; TEST FPS
14062 071140 142100                    .WORD   142100          ;RESULT FPS
14063 071142 000012                    .WORD   12              ;FEC
14064
14065 071144 012737 000001 003030      ;13/OVERFLOW. TRAP
14066 071152 004767 000116              MOV      #1,#FLAG          ;INTERRUPT
14067 071156 177777 177777              JSR      R7,MLFSUB        ;DO TEST
14068 071162 040300 000000              .WORD   177777,-1       ;ACO
14069 071166 100077 177777              .WORD   40300.0         ;FSRC
14070 071172 001117                    .WORD   100077,-1       ;RESULT
14071 071174 101116                    .WORD   1117           ; TEST FPS
14072 071176 000010                    .WORD   101116          ;RESULT FPS
14073
14074 071200 012737 000002 003030      ;14/OVERFLOW NO TRAP
14075 071206 004767 000062              MOV      #2,#FLAG          ;NO INTERRUPT
14076 071212 077700 000000              JSR      R7,MLFSUB        ;DO TEST
14077 071216 077700 000000              .WORD   77700.0         ;ACO
14078 071222 000000 000000              .WORD   77700.0         ;FSRC
14079 071226 040117                    .WORD   0.0            ;RESULT
14080 071230 040106                    .WORD   40117          ; TEST FPS
14081 071232 000010                    .WORD   40106          ;RESULT FPS
14082
14083 071234 012737 000001 003030      ;15/UNDEFINED VARIABLE IN FSRC, TRAP ENABLED
14084 071242 004767 000026              MOV      #1,#FLAG          ;INTERRUPT
14085 071246 123465 000000              JSR      R7,MLFSUB        ;DO TEST
14086 071252 100022 000000              .WORD   123465.0        ;ACO
14087 071256 123465 000000              .WORD   100022.0        ;FSRC
14088 071262 004000                    .WORD   123465.0        ;RESULT
14089 071264 104000                    .WORD   4000           ; TEST FPS
14090 071266 000014                    .WORD   104000          ;RESULT FPS
14091
14092 071270 000167 000242              .WORD   14             ;FEC
14093
14094
14095
14096
14097
14098
14099
14100
14101
14102
14103
14104
14105 071274 012605                    ;
14106 071276 012737 071356 000244      MLFSUB: MOV      (SP)+,R5    ; RETURN ADDRESS TO USE AS POINTER
14107 071304 012702 000200              MOV      #50,#FPVEC       ;REDIRECT TRAP VECTOR
14108 071310 170102                    MOV      #200,R2          ;SET TO DOUBLE MODE FOR LOAD
14109 071312 172415                    LDFPS   R2               ;LOAD FPS
14110 071314 010504                    LDD     (R5),ACO         ;LOAD ACO WITH TEST DATA
14111 071316 062704 000004              MOV      R5,R4           ;POINT TO FSRC DATA
14112 071322 016502 000014              ADD     #4,R4
14112 071322 016502 000014              MOV     14(R5),R2        ;GET TEST FPS

```

FLOATING POINT TESTS

```

14113 071326 170102          LDFPS  R2          ;LOAD TEST FPS
14114                      ;
14115 071330 171014          MULF   (R4),ACO     ;*TEST INSTRUCTION
14116 071332 170001          1$:   SETF           ;WAIT FOR POSSIBLE FPA TRAP.
14117                      ;
14118                      ;INSTRUCTION DIDNT TRAP
14119                      ;
14120 071334 032737 000001 003030  BIT    #1,#FLAG      ;VERIFY A NO TRAP CONDITION
14121 071342 001426          BEQ    2$           ;BRANCH IF GOOD
14122 071344 004737 140132    CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14123 071350 104003          ERROR  +3          ;FPP ERROR
14124                      ;INSTRUCTION SHOULD HAVE TRAPPED
14125 071352 000167 000042    JMP    2$           ;REJOIN CODE
14126                      ;
14127                      ;INSTRUCTION TRAPPED
14128                      ;
14129 071356 032737 000001 003030 50$:  BIT    #1,#FLAG      ;SEE IF EXPECTING A TRAP
14130 071364 001005          BNE   51$          ;BRANCH IF EXPECTING A TRAP
14131 071366 004737 140132    CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14132 071372 104003          ERROR  +3          ;FPP ERROR
14133                      ;INSTRUCTION WASNT SUPPOSE TO TRAP
14134 071374 000167 000020    JMP    2$           ;REJOIN CODE
14135 071400 012604          51$:  MOV    (SP)+,R4    ;SEE IF PC = INSTRUCTION
14136 071402 005726          TST   (SP)+        ;CLEAN UP STACK
14137 071404 022704 071332    CMP   #1$,R4       ;
14138 071410 001403          BEQ   2$           ;BRANCH IF GOOD COMPARE
14139 071412 004737 140132    CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14140 071416 104003          ERROR  +3          ;FPP ERROR
14141                      ;PC WAS INCORRECT
14142                      ;
14143                      ;COMMON CODE FOR TRAP AND NO TRAP
14144                      ;
14145 071420 170203          2$:   STFPS  R3          ;SAVE FPS
14146 071422 012702 000200    MOV   #200,R2      ;SET FPP TO DOUBLE
14147 071426 170102          LDFPS  R2
14148 071430 012701 003142    MOV   @RECDST,R1   ;POINT TO RECEIVED DATA TABLE
14149 071434 174011          STD   ACO,(R1)     ;SAVE ACO RESULT
14150 071436 026503 000016    CMP   16(R5),R3    ;VERIFY STATUS
14151 071442 001403          BEQ   3$           ;BRANCH IF GOOD
14152 071444 004737 140132    CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14153 071450 104003          ERROR  +3          ;FPP ERROR
14154                      ;BAD FPS
14155 071452 010504          3$:   MOV   R5,R4          ;POINT TO EXPECTED DATA
14156 071454 062704 000010    ADD   #10,R4
14157 071460 004767 046376    4$:   JSR   R7,DATVFR    ;VERIFY DATA
14158 071464 005767 111430    TST   COUNT
14159 071470 001403          BEQ   5$           ;BRANCH IF GOOD
14160 071472 004737 140132    CALL   @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14161 071476 104003          ERROR  +3          ;FPP ERROR
14162                      ;BAD ACO
14163 071500 005737 003030    5$:   TST   @FLAG      ;SEE IF NEED TO CHECK FEC
14164 071504 001002          BNE   6$           ;BRANCH IF NEED TO CHECK
14165 071506 000165 000020    JMP   20(R5)       ;RETURN FROM TEST
14166                      ;
14167                      ;VERIFY ERROR STATUS
14168                      ;
14169 071512 170301          6$:   STST  R1          ;SAVE FEC

```

FLOATING POINT TESTS

```

14170 071514 016504 000020      MOV      20(R5),R4      ;GET FEC
14171 071520 020401      CMP      R4,R1        ;VERIFY FEC
14172 071522 001403      BEQ      7$          ;BRANCH IF GOOD
14173 071524 004737 140132      CALL     @DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14174 071530 104003      ERROR   +3          ;FPP ERROR
14175                                ;BAD FEC
14176 071532 000165 000022      7$:      JMP      22(R5)    ;RETURN FROM TEST
14177 071536                                HOP12:
14180                                ;
14181                                ;-----
14182                                ;TEST MULD
14183                                ;
14184 071536                                MMULD:
14185                                ;
14186                                ;1/AC=0
14187 071536 005037 003030      CLR      @FLAG        ;NO INTERRUPT
14188 071542 004767 000554      JSR      R7,MLDSUB    ;DO TEST
14189 071546 000100 000000 000000      .WORD   100,0,0,0    ;ACO
14190 071556 000411 177777 000000      .WORD   411,-1,0,1   ;FSRC
14191 071566 000000 000000 000000      .WORD   0,0,0,0     ;RESULT
14192 071576 000200      .WORD   200          ; TEST FPS
14193 071600 000204      .WORD   204          ;RESULTANT FPS
14194                                ;2/FSRC=0
14195 071602 005037 003030      CLR      @FLAG        ;NO INTERRUPT
14196 071606 004767 000510      JSR      R7,MLDSUB    ;DO TEST
14197 071612 077777 000000 000000      .WORD   77777,0,0,0 ;ACO
14198 071622 000000 000000 000000      .WORD   0,0,0,0     ;FSRC
14199 071632 000000 000000 000000      .WORD   0,0,0,0     ;RESULT
14200 071642 007700      .WORD   7700         ; TEST FPS
14201 071644 007704      .WORD   7704         ;RESULTANT FPS
14202                                ;3/AC=1
14203 071646 005037 003030      CLR      @FLAG        ;NO INTERRUPT
14204 071652 004767 000444      JSR      R7,MLDSUB    ;DO TEST
14205 071656 040200 000000 000000      .WORD   40200,0,0,0 ;ACO
14206 071666 000277 177777 177777      .WORD   277,-1,-1,-1 ;FSRC
14207 071676 000277 177777 177777      .WORD   277,-1,-1,-1 ;RESULT
14208 071706 007717      .WORD   7717         ; TEST FPS
14209 071710 007700      .WORD   7700         ;RESULTANT FPS
14210                                ;4/AC>FSRC>0, TRUNCATE
14211 071712 005037 003030      CLR      @FLAG        ;NO INTERRUPT
14212 071716 004767 000400      JSR      R7,MLDSUB    ;DO TEST
14213 071722 065500 000000 000000      .WORD   65500,0,0,1 ;ACO
14214 071732 037577 177777 177777      .WORD   37577,-1,-1,-2 ;FSRC
14215 071742 065077 177777 177777      .WORD   65077,-1,-1,-1 ;RESULT
14216 071752 007717      .WORD   7717         ; TEST FPS

```

FLOATING POINT TESTS

```

14217 071754 007700          .WORD 7700          ;RESULTANT FPS
14218                               ;5/AC<FSRC<0
14219 071756 005037 003030  CLR  @FLAG          ;NO INTERRUPT
14220 071762 004767 000334  JSR  R7,MLDSUB      ;DO TEST
14221 071765 005577 177777 177777 .WORD 137577,-1,-1,-2 ;ACO
071774 177776
14222 071776 165400 000000 000000 .WORD 165400,0,0,1   ;FSRC
072004 000001
14223 072006 065000 000000 000000 .WORD 65000,0,0,0   ;RESULT
072014 000000
14224 072016 007717          .WORD 7717          ; TEST FPS
14225 072020 007700          .WORD 7700          ;RESULTANT FPS
14226                               ;6/AC>FSRC>0
14227 072022 005037 003030  CLR  @FLAG          ;NO INTERRUPT
14228 072026 004767 000270  JSR  R7,MLDSUB      ;DO TEST
14229 072032 017500 000000 000000 .WORD 17500,0,0,0   ;ACO
072040 000000
14230 072042 123652 125252 125252 .WORD 123652,125252,125252,125252 ;FSRC
072050 125252
14231 072052 103177 177777 177777 .WORD 103177,-1,-1,-1 ;RESULT
072060 177777
14232 072062 000200          .WORD 200           ; TEST FPS
14233 072064 000210          .WORD 210           ;RESULTANT FPS
14234                               ;7/UNDERFLOW, TRAPS DISABLED
14235 072066 005037 003030  CLR  @FLAG          ;NO INTERRUPT
14236 072072 004767 000224  JSR  R7,MLDSUB      ;DO TEST
14237 072076 000300 000000 000000 .WORD 300,0,0,252   ;ACO
072104 000252
14238 072106 000377 000001 000002 .WORD 377,1,2,3     ;FSRC
072114 000003
14239 072116 000000 000000 000000 .WORD 0,0,0,0       ;RESULT
072124 000000
14240 072126 005740          .WORD 5740          ; TEST FPS
14241 072130 005744          .WORD 5744          ;RESULT FPS
14242                               ;8/UNDERFLOW, TRAP ENABLED
14243 072132 012737 000001 003030  MOV  #1,@FLAG       ;INTERRUPT
14244 072140 004767 000156  JSR  R7,MLDSUB      ;DO TEST
14245 072144 100277 000001 000002 .WORD 100277,1,2,-1 ;ACO
072152 177777
14246 072154 100300 000001 000001 .WORD 100300,1,1,1   ;FSRC
072162 000001
14247 072164 040417 040001 077403 .WORD 40417,40001,77403,0 ;RESULT
072172 000000
14248 072174 002217          .WORD 2217          ; TEST FPS
14249 072176 102200          .WORD 102200        ;RESULT FPS
14250 072200 000012          .WORD 12            ;FEC
14251                               ;9/OVERFLOW, TRAPS DISABLED
14252 072202 005037 003030  CLR  @FLAG          ;NO INTERRUPT
14253 072206 004767 000110  JSR  R7,MLDSUB      ;DO TEST
14254 072212 177777 177777 177777 .WORD -1,-1,1,-1    ;ACO
072220 177777
14255 072222 040200 177777 177777 .WORD 40200,1,-1,-1 ;FSRC
072230 177777
14256 072232 000000 000000 000000 .WORD 0,0,0,0       ;RESULT
072240 000000
14257 072242 006740          .WORD 6740          ; TEST FPS
14258 072244 006746          .WORD 6746          ;RESULT FPS
    
```


FLOATING POINT TESTS

```

14259                                     ;10/OVERFLOW, TRAPS ENABLED
14260 072246 012737 000001 003030      MOV     #1,0#FLAG                      ;INTERRUPT
14261 072254 004767 000042              JSR     R7,MLDSUB                      ;DO TEST
14262 072260 157700 025252 025252      .WORD  157700,25252,25252,25252      ;ACO
      U/2266 025252
14263 072270 167700 000000 000000      .WORD  167700,0,0,0                  ;FSRC
      072276 000000
14264 072300 007420 017777 117777      .WORD  7420,017777,117777,117777    ;RESULT
      072306 117777
14265 072310 001240                      .WORD  1240                          ; TEST FPS
14266 072312 101242                      .WORD  101242                       ;RESULT FPS
14267 072314 000010                      .WORD  10                            ;FEC
14268                                     ;
14269 072316 000167 000242              JMP     HOP13
14270                                     ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14271                                     ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14272                                     ;
14273                                     ;               ACO
14274                                     ;               FSRC
14275                                     ;               FPS BEFORE EXECUTION
14276                                     ;               FPS AFTER EXECUTION
14277                                     ;               (FEC)
14278                                     ;
14279                                     ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14280                                     ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14281                                     ;
14282 072322 012605                      MLDSUB: MOV   (SP)+,R5                 ; RETURN ADDRESS TO USE AS POINTER
14283 072324 012737 072404 000244      MOV     #50,0#FPVEC                  ;REDIRECT TRAP VECTOR
14284 072332 012702 000200              MOV     #200,R2                      ;SET TO DOUBLE MODE FOR LOAD
14285 072336 170102                      LDFPS  R2                            ;LOAD FPS
14286 072340 172415                      LDD    (R5),ACO                      ;LOAD ACO WITH TEST DATA
14287 072342 010501                      MOV     R5,R1                        ;POINT TO FSRC DATA
14288 072344 062701 000010              ADD     #10,R1
14289 072350 016502 000030              MO     30(R5),R2                    ;GET TEST FPS
14290 072354 170102                      LDFPS  R2                            ;LOAD TEST FPS
14291                                     ;
14292 072356 171011                      ;     MULD (R1),ACO                 ;*TEST INSTRUCTION
14293 072360 170011      1$:     SETD                      ;WAIT FOR POSSIBLE FPA TRAP.
14294                                     ;
14295                                     ;INSTRUCTION DIDNT TRAP
14296                                     ;
14297 072362 032737 000001 003030      BIT     #1,0#FLAG                    ;VERIFY A NO TRAP CONDITION
14298 072370 001426                      BEQ     2$                            ;BRANCH IF GOOD
14299 072372 004737 140132              CALL   0#DETFPA                      ;DETERMINE FLOATING POINT FAULT. $$$
14300 072376 104003                      ERROR   +3                            ;FPP ERROR
14301                                     ;INSTRUCTION SHOULD HAVE TRAPPED
14302 072400 000167 000042              JMP     2$                            ;REJOIN CODE
14303                                     ;
14304                                     ;INSTRUCTION TRAPPED
14305                                     ;
14306 072404 032737 000001 003030      50$:   BIT     #1,0#FLAG              ;SEE IF EXPECTING A TRAP
14307 072412 001005                      BNE     51$                            ;BRANCH IF EXPECTING A TRAP
14308 072414 004737 140132              CALL   0#DETFPA                      ;DETERMINE FLOATING POINT FAULT. $$$
14309 072420 104003                      ERROR   +3                            ;FPP ERROR
14310                                     ;INSTRUCTION WASNT SUPPOSE TO TRAP
14311 072422 000167 000020              JMP     2$                            ;REJOIN CODE
14312 072426 012604      51$:   MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
    
```

FLOATING POINT TESTS

```

14313 072430 005726          TST      (SP)+          ;CLEAN UP STACK
14314 072432 022704 072360    CMP      #1$,R4        ;
14315 072436 001403          BEQ      2$            ;BRANCH IF GOOD COMPARE
14316 072440 004737 140132    CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14317 072444 104003          ERROR   +3            ;FPP ERROR
14318                                ;PC WAS INCORRECT
14319
14320                                ;COMMON CODE FOR TRAP AND NO TRAP
14321                                ;
2$:      STFPS    R3          ;SAVE FPS
        MOV      #200,R2      ;SET FPP TO DOUBLE
        LDFPS    R2
        MOV      #RECDST,R1   ;POINT TO RECEIVED DATA TABLE
        STD      ACO,(R1)     ;SAVE ACO RESULT
        CMP      32(R5),R3    ;VERIFY STATUS
        BEQ      3$          ;BRANCH IF GOOD
        CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
        ERROR   +3            ;FPP ERROR
14322 072446 170203          ;BAD FPS
14323 072450 012702 000200    MOV      #200,R2      ;POINT TO EXPECTED DATA
14324 072454 170102          ADD      #20,R4
14325 072456 012701 003142    JSR      R7,DATVER     ;VERIFY DATA
14326 072462 174011          TST      COUNT
14327 072464 026503 000032    BEQ      5$          ;BRANCH IF GOOD
14328 072470 001403          CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14329 072472 004737 140132    ERROR   +3            ;FPP ERROR
14330 072476 104003          ;BAD ACO
14331
14332 072500 010504          5$:      TST      @#FLAG    ;SEE IF NEED TO CHECK FEC
14333 072502 062704 000020    BNE      6$          ;BRANCH IF NEED TO CHECK
14334 072506 004767 045366    JMP      34(R5)       ;RETURN FROM TEST
14335 072512 005767 110402    ;VERIFY ERROR STATUS
14336 072516 001403          6$:      STST     R1          ;SAVE FEC
14337 072520 004737 140132    MOV      34(R5),R4    ;GET FEC
14338 072524 104003          CMP      R4,R1        ;VERIFY FEC
14339
14340 072526 005737 003030    BEQ      7$          ;BRANCH IF GOOD
14341 072532 001002          CALL    @#DETFPA      ;DETERMINE FLOATING POINT FAULT. $$$
14342 072534 000165 000034    ERROR   +3            ;FPP ERROR
14343                                ;BAD FEC
14344 072540 170301          7$:      JMP      36(R5)    ;RETURN FROM TEST
14345 072542 016504 000034    HOP13:
14346 072546 020401          ;
14347 072550 001403          ;-----
14348 072552 004737 140132    ;TEST MODF
14349 072556 104003          ;
14350                                ;MMODF:
14351 072560 000165 000036    ;
14352 072564          ;1/AC=0 FSRC=0
14353                                ;
14354                                ;NO INTERRUPT
14355                                ;DO TEST
14356                                ;
14357                                ;FSRC
14358                                ;FRACTIONAL RESULT
14359 072564          CLR      @#FLAG
14360                                ;INTEGER RESULT
14361                                ;TEST FPS
14362 072564 005037 003030    JSR      R7,MDFSUB
14363 072570 004767 000554    .WORD   100,0 ;ACO
14364 072574 000100 000000    .WORD   12346,-1
14365 072600 012346 177777    .WORD   0,0
14366 072604 000000 000000    .WORD   0,0
14367 072610 000000 000000    .WORD   13
14368 072614 000013          .WORD   4
14369 072616 000004          ;RESULTANT FPS
14370                                ;
14371 072620 005037 003030    CLR      @#FLAG      ;NO INTERRUPT

```

FLOATING POINT TESTS

```

14372 072624 004767 000520      JSR      R7,MDFSUB      ;DO TEST
14373 072630 012356 177777      .WORD   12356,-1      ;ACO
14374 072634 000000 000000      .WORD   0,0          ;FSRC
14375 072640 000000 000000      .WORD   0,0          ;FRACTIONAL RESULT
14376 072644 000000 000000      .WORD   0,0          ;INTEGER RESULT
14377 072650 000003          .WORD   3            ;TEST FPS
14378 072652 000004          .WORD   4            ;RESULTANT FPS
14379                                ;3/AC=0
14380 072654 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14381 072660 004767 000464      JSR      R7,MDFSUB      ;DO TEST
14382 072664 000000 000000      .WORD   0,0          ;ACO
14383 072670 177777 177777      .WORD   -1,-1        ;FSRC
14384 072674 000000 000000      .WORD   0,0          ;FRACTIONAL RESULT
14385 072700 000000 000000      .WORD   0,0          ;INTEGER RESULT
14386 072704 007500          .WORD   7500         ;TEST FPS
14387 072706 007504          .WORD   7504         ;RESULT FPS
14388                                ;4/AC>FSRC>0
14389 072710 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14390 072714 004767 000430      JSR      R7,MDFSUB      ;DO TEST
14391 072720 046252 125252      .WORD   46252,125252 ;ACO
14392 072724 040300 000000      .WORD   40300,0      ;FSRC
14393 072730 000000 000000      .WORD   0,0          ;FRACTIONAL RESULT
14394 072734 046377 177777      .WORD   46377,-1    ;INTEGER RESULT
14395 072740 000013          .WORD   13           ;TEST FPS
14396 072742 000004          .WORD   4            ;RESULTANT FPS
14397                                ;5/AC>FSRC>0
14398 072744 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14399 072750 004767 000374      JSR      R7,MDFSUB      ;DO TEST
14400 072754 077652 125252      .WORD   77652,125252 ;ACO
14401 072760 040300 000000      .WORD   40300,0      ;FSRC
14402 072764 000000 000000      .WORD   0,0          ;FRACTIONAL RESULT
14403 072770 077777 177777      .WORD   77777,-1    ;INTEGER RESULT
14404 072774 000000          .WORD   0            ;TEST FPS
14405 072776 000004          .WORD   4            ;RESULTANT FPS
14406                                ;6/AC>0<FSRC, INTEGERS
14407 073000 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14408 073004 004767 000340      JSR      R7,MDFSUB      ;DO TEST
14409 073010 060600 000000      .WORD   60600,0      ;ACO
14410 073014 147400 025700      .WORD   147400,25700 ;FSRC
14411 073020 000000 000000      .WORD   0,0          ;FRACTIONAL RESULT
14412 073024 170000 025700      .WORD   170000,25700 ;INTEGER RESULT
14413 073030 007400          .WORD   7400         ;TEST FPS
14414 073032 007404          .WORD   7404         ;RESULT FPS
14415                                ;7/AC<0<FSRC, FRACTIONAL
14416 073034 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14417 073040 004767 000304      JSR      R7,MDFSUB      ;DO TEST
14418 073044 100227 177777      .WORD   100227,-1    ;ACO
14419 073050 044025 025252      .WORD   44025,25252  ;FSRC
14420 073054 104061 021251      .WORD   104061,21251 ;FRACTIONAL RESULT
14421 073060 000000 000000      .WORD   0,0          ;INTEGER RESULT
14422 073064 000000          .WORD   0            ;TEST FPS
14423 073066 000010          .WORD   10           ;RESULT FPS
14424                                ;8/AC<0>FSRC, TRUNCATE
14425 073070 005037 003030      CLR      @#FLAG        ;NO INTERRUPT
14426 073074 004767 000250      JSR      R7,MDFSUB      ;DO TEST
14427 073100 046252 125252      .WORD   46252,125252 ;ACO
14428 073104 040300 000000      .WORD   40300,0      ;FSRC

```

FLOATING POINT TESTS

14429	073110	000000	000000	.WORD	0,0	;FRACTIONAL RESULT
14430	073114	046377	177777	.WORD	46377,-1	;INTEGER RESULT
14431	073120	000053		.WORD	53	; TEST FPS
14432	073122	000044		.WORD	44	;RESULT FPS
14433				;9/ROUND INTEGER		
14434	073124	005037	003030	CLR	0#FLAG	;NO INTERRUPT
14435	073130	004767	000214	JSR	R7,MDFSUB	;DO TEST
14436	073134	046252	125252	.WORD	46252,125252	;ACO
14437	073140	040300	000000	.WORD	40300,0	;FSRC
14438	073144	000000	000000	.WORD	0,0	;FRACTIONAL RESULT
14439	073150	046377	177777	.WORD	46377,-1	;INTEGER RESULT
14440	073154	000013		.WORD	13	; TEST FPS
14441	073156	000004		.WORD	4	;RESULT FPS
14442				;10/TRUNCATE FRACTION		
14443	073160	005037	003030	CLR	0#FLAG	;NO INTERRUPT
14444	073164	004767	000160	JSR	R7,MDFSUB	;DO TEST
14445	073170	040777	177777	.WORD	40777,-1	;ACO
14446	073174	040200	000000	.WORD	40200,0	;FSRC
14447	073200	040177	177770	.WORD	40177,177770	;FRACTIONAL RESULT
14448	073204	040740	000000	.WORD	40740,0	;INTEGER RESULT
14449	073210	000000		.WORD	0	; TEST FPS
14450	073212	000000		.WORD	0	;RESULT FPS
14451				;11/ROUND INTEGER		
14452	073214	005037	003030	CLR	0#FLAG	;NO INTERRUPT
14453	073220	004767	000124	JSR	R7,MDFSUB	;DO TEST
14454	073224	000000	000000	.WORD	0,0	;ACO
14455	073230	000000	000000	.WORD	0,0	;FSRC
14456	073234	000000	000000	.WORD	0,0	;FRACTIONAL RESULT
14457	073240	000000	000000	.WORD	0,0	;INTEGER RESULT
14458	073244	000000		.WORD	0	; TEST FPS
14459	073246	000004		.WORD	4	;RESULT FPS
14460				;12/ROUND FRACTION		
14461	073250	005037	003030	CLR	0#FLAG	;NO INTERRUPT
14462	073254	004767	000070	JSR	R7,MDFSUB	;DO TEST
14463	073260	040225	125252	.WORD	40225,125252	;ACO
14464	073264	066652	052525	.WORD	66652,52525	;FSRC
14465	073270	000000	000000	.WORD	0,0	;FRACTIONAL RESULT
14466	073274	066707	025160	.WORD	66707,25160	;INTEGER RESULT
14467	073300	007027		.WORD	7027	; TEST FPS
14468	073302	007004		.WORD	7004	;RESULT FPS
14469				;/OVERFLOW		
14470	073304	012737	000001 003030	MOV	#1,0#FLAG	;INTERRUPT
14471	073312	004767	000032	JSR	R7,MDFSUB	;DO TEST
14472	073316	076000	000000	.WORD	76000,0	;ACO
14473	073322	076000	000000	.WORD	76000,0	;FSRC
14474	073326	000000	000000	.WORD	0,0	;FRACTIONAL RESULT
14475	073332	033600	000000	.WORD	33600,0	;INTEGER RESULT
14476	073336	001000		.WORD	1000	; TEST FPS
14477	073340	101006		.WORD	101006	;RESULT FPS
14478	073342	000010		.WORD	10	;FEC
14479				;		
14480	073344	000167	000310	JMP	HOP14	
14481				;		
14482				;*****		
14483				;*****		
14484				;		
14485				; ACO		

FLOATING POINT TESTS

```

14486      ;          FSRC
14487      ;          FRACTIONAL RESULT
14488      ;          INTEGER RESULT
14489      ;          FPS BEFORE EXECUTION
14490      ;          FPS AFTER EXECUTION
14491      ;          (FEC)
14492      ;
14493      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14494      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14495      ;
14496 073350 012605      MDFSUB: MOV      (SP)+,R5          ; RETURN ADDRESS TO USE AS POINTER
14497 073352 012737 073440 000244  MOV      #50#,R5          ; REDIRECT TRAP VECTOR
14498 073360 012702 000200      MOV      #200,R2         ; SET TO DOUBLE MODE FOR LOAD
14499 073364 170102      LDFPS   R2              ; LOAD FPS
14500 073366 172415      LDD     (R5),AC0        ; LOAD AC0 WITH TEST DATA
14501 073370 012701 073650      MOV      #MODGAR,R1     ; LOAD KNOWN INTO AC1
14502 073374 172511      LDD     (R1),AC1        ;
14503 073376 010501      MOV      R5,R1          ; POINT TO FSRC DATA
14504 073400 062701 000004      ADD     #4,R1           ;
14505 073404 016502 000020      MOV      20(R5),R2     ; GET TEST FPS
14506 073410 170102      LDFPS   R2              ; LOAD TEST FPS
14507      ;
14508 073412 171411      MODF   (R1),AC0        ; *TEST INSTRUCTION
14509 073414 170001      1$:   SETF              ; WAIT FOR POSSIBLE FPA TRAP.
14510      ;
14511      ; INSTRUCTION DIDNT TRAP
14512      ;
14513 073416 032737 000001 003030      BIT     #1,#FLAG        ; VERIFY A NO TRAP CONDITION
14514 073424 001426      BEQ     2$              ; BRANCH IF GOOD
14515 073426 004737 140132      CALL   #DETFPA         ; DETERMINE FLOATING POINT FAULT. $$$
14516 073432 104003      ERROR  +3              ; FPP ERROR
14517      ; INSTRUCTION SHOULD HAVE TRAPPED
14518 073434 000167 000042      JMP     2$              ; REJOIN CODE
14519      ;
14520      ; INSTRUCTION TRAPPED
14521      ;
14522 073440 032737 000001 003030      50$:   BIT     #1,#FLAG        ; SEE IF EXPECTING A TRAP
14523 073446 001005      BNE    51$              ; BRANCH IF EXPECTING A TRAP
14524 073450 004737 140132      CALL   #DETFPA         ; DETERMINE FLOATING POINT FAULT. $$
14525 073454 104003      ERROR  +3              ; FPP ERROR
14526      ; INSTRUCTION WASNT SUPPOSE TO TRAP
14527 073456 000167 000020      JMP     2$              ; REJOIN CODE
14528 073462 012604      51$:   MOV      (SP)+,R4        ; SEE IF PC = INSTRUCTION
14529 073464 005726      TST    (SP)+           ; CLEAN UP STACK
14530 073466 022704 073414      CMP     #1#,R4          ;
14531 073472 001403      BEQ     2$              ; BRANCH IF GOOD COMPARE
14532 073474 004737 140132      CALL   #DETFPA         ; DETERMINE FLOATING POINT FAULT. $$$
14533 073500 104003      ERROR  +3              ; FPP ERROR
14534      ; PC WAS INCORRECT
14535      ;
14536      ; COMMON CODE FOR TRAP AND NO TRAP
14537      ;
14538 073502 170203      2$:   STFPS   R3          ; SAVE FPS
14539 073504 012702 000200      MOV      #200,R2         ; SET FPP TO DOUBLE
14540 073510 170102      LDFPS   R2              ;
14541 073512 012701 003142      MOV      #RECDST,R1     ; POINT TO RECEIVED DATA TABLE
14542      ; SAVE FRACTIONAL RESULT

```

FLOATING POINT TESTS

```

14543 073516 174011          STD      ACO,(R1)          ;SAVE ACO RESULT
14544 073520 026503 000022    CMP      22(R5),R3        ;VERIFY STATUS
14545 073524 001403          BEQ      3$              ;BRANCH IF GOOD
14546 073526 004737 140132    CALL     @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
14547 073532 104003          ERROR    +3              ;FPP ERROR
14548                                ;BAD FPS
14549 073534 010504          3$:     MOV      R5,R4          ;POINT TO EXPECTED DATA
14550 073536 062704 000010    ADD     @10,R4
14551 073542 004767 044314          4$:     JSR      R7,DATVFR        ;VERIFY DATA
14552 073546 005767 107346    TST     COUNT
14553 073552 001403          BEQ      5$              ;BRANCH IF GOOD
14554 073554 004737 140132    CALL     @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
14555 073560 104003          ERROR    +3              ;FPP ERROR
14556                                ;BAD ACO
14557                                ;SAVE INTEGER RESULT
14558 073562 174111          5$:     STD      AC1,(R1)        ;SAVE AC1 RESULT
14559 073564 010504          MOV      R5,R4          ;POINT TO EXPECTED
14560 073566 062704 000014    ADD     @14,R4
14561 073572 004767 044264          JSR      R7,DATVFR        ;VERIFY DATA
14562 073576 005767 107316    TST     COUNT
14563 073602 001403          BEQ      6$              ;BRANCH IF GOOD
14564 073604 004737 140132    CALL     @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
14565 073610 104003          ERROR    +3              ;FPP ERROR
14566                                ;BAD AC1
14567 073612 005737 003030          6$:     TST     @FLAG          ;SEE IF NEED TO CHECK FEC
14568 073616 001002          BNE     7$              ;BRANCH IF NEED TO CHECK
14569 073620 000165 000024    JMP     24(R5)          ;RETURN FROM TEST
14570 073624 170301          7$:     STST    R1          ;SAVE FEC
14571 073626 016504 000024    MOV     24(R5),R4       ;GET FEC
14572 073632 020401          CMP     R4,R1          ;VERIFY FEC
14573 073634 001403          BEQ     8$              ;BRANCH IF GOOD
14574 073636 004737 140132    CALL     @DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
14575 073642 104003          ERROR    +3              ;FPP ERROR
14576                                ;BAD FEC
14577 073644 000165 000026          8$:     JMP     26(R5)          ;RETURN FROM TEST
14578                                ;
14579 073650 177777 177777 177777 MODGAR: .WORD -1,-1,-1,-1 ;KNOWN DATA FOR AC1
14580 073656 177777
14581 073660
14582                                HOP14:
14583                                ;
14584                                ;-----
14585                                ;TEST MODD
14586                                ;
14587                                ;MMODD:
14588 073660
14589                                ;
14590                                ;1/AC>FSRC=0
14591 073660 005037 003030          CLR     @FLAG          ;NO INTERRUPT
14592 073664 004767 001164          JSR     R7,MDDSUB        ;DO TEST
14593 073670 012345 177777 177777 .WORD 12345,-1,-1,-1 ;ACO
14594 073676 177777
14595 073700 000100 000000 000000 .WORD 100,0,0,0 ;FSRC
14596 073706 000000
14597 073710 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
14598 073716 000000
14599 073720 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
14600 073726 000000

```

FLOATING POINT TESTS

```

14597 073730 000200          .WORD 200          ; TEST FPS
14598 073732 000204          .WORD 204          ; RESULTANT FPS
14599                          ;2/AC<0<FSRC
14600 073734 005037 003030    CLR          @FLAG      ;NO INTERRUPT
14601 073740 004767 001110    JSR          R7,MDDSUB   ;DO TEST
14602 073744 000000 000000 000000 .WORD 0,0,0,0 ;ACO
14603 073754 001234 177777 000000 .WORD 1234,-1,0,0 ;FSRC
14604 073762 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
14605 073772 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
14606 074004 007717          .WORD 7717         ; TEST FPS
14607 074006 007704          .WORD 7704         ; RESULTANT FPS
14608                          ;3/AC>FSRC>0
14609 074010 005037 003030    CLR          @FLAG      ;NO INTERRUPT
14610 074014 004767 001034    JSR          R7,MDDSUB   ;DO TEST
14611 074020 056252 125252 125252 .WORD 56252,125252,125252,125250 ;ACO
14612 074026 125250 000000 000000 .WORD 40300,0,0,0 ;FSRC
14613 074030 040300 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
14614 074040 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
14614 074050 056377 177777 177777 .WORD 56377,-1,-1,-4
14615 074056 177774          .WORD 213          ; TEST FPS
14616 074060 000213          .WORD 204          ; RESULTANT FPS
14617                          ;4/AC<0>FSRC
14618 074064 005037 003030    CLR          @FLAG      ;NO INTERRUPT
14619 074070 004767 000760    JSR          R7,MDDSUB   ;DO TEST
14620 074074 140240 000000 000000 .WORD 140240,0,0,0 ;ACO
14621 074102 000000 000000 000000 .WORD 63714,146314,133572,167737 ;FSRC
14621 074104 063714 146314 133572 .WORD 0,0,0,0 ;FRACTIONAL RESULT
14622 074112 167737 000000 000000 .WORD 163777,-1,162531,125726 ;INTEGER RESULT
14622 074114 000000 000000 000000 .WORD 163777,-1,162531,125726
14623 074122 000000 000000 000000 .WORD 210          ; TEST FPS
14623 074124 163777 177777 162531 .WORD 204          ; RESULTANT FPS
14624 074132 125726          .WORD 210          ; TEST FPS
14625 074134 000210          .WORD 204          ; RESULTANT FPS
14626                          ;5/AC>FSRC>0
14627 074140 005037 003030    CLR          @FLAG      ;NO INTERRUPT
14628 074144 004767 000704    JSR          R7,MDDSUB   ;DO TEST
14629 074150 056200 000000 000000 .WORD 56200,0,0,1 ;ACO
14630 074156 000001 000000 000000 .WORD 40340,0,0,0 ;FSRC
14631 074160 040340 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
14631 074166 000000 000000 000000 .WORD 56340,0,0,1 ;INTEGER RESULT
14632 074170 000000 000000 000000 .WORD 56340,0,0,1
14632 074176 000000 000000 000000 .WORD 213          ; TEST FPS
14633 074200 056340 000000 000000 .WORD 204          ; RESULTANT FPS
14634 074206 000001 000000 000000 .WORD 213          ; TEST FPS
14635                          ;6/TRUNCATE
14635 074210 000213          .WORD 204          ; RESULTANT FPS
14636 074212 000204          .WORD 213          ; TEST FPS
14637 074214 005037 003030    CLR          @FLAG      ;NO INTERRUPT
14637 074220 004767 000630    JSR          R7,MDDSUB   ;DO TEST

```

FLOATING POINT TESTS

```

14638 074224 056252 125252 125252 .WORD 56252,125252,125252,125252 ;ACO
      074232 125252
14639 074234 040300 000000 000000 .WORD 40300,0,0,0 ;FSRC
      074242 000000
14640 074244 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
      074252 000000
14641 074254 056377 177777 177777 .WORD 56377,-1,-1,1 ;INTEGER RESULT
      074262 177777
14642 074264 000253 .WORD 253 ; TEST FPS
14643 074266 000244 .WORD 244 ;RESULT FPS
14644 ;7/TRUNCATE FRACTION
14645 074270 005037 003030 CLR @FLAG ;NO INTERRUPT
14646 074274 004767 000554 JSR R7,MDDSUB ;DO TEST
14647 074300 023252 125252 125252 .WORD 23252,125252,125252,125252 ;ACO
      074306 125252
14648 074310 040300 000000 000000 .WORD 40300,0,0,0 ;FSRC
      074316 000000
14649 074320 023377 177777 177777 .WORD 23377,1,-1,-1 ;FRACTIONAL RESULT
      074326 177777
14650 074330 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
      074336 000000
14651 074340 000253 .WORD 253 ; TEST FPS
14652 074342 000240 .WORD 240 ;RESULT FPS
14653 ;8/ROUND INTEGER
14654 074344 005037 003030 CLR @FLAG ;NO INTERRUPT
14655 074350 004767 000500 JSR R7,MDDSUB ;DO TEST
14656 074354 076600 000000 000000 .WORD 76600,0,0,125252 ;ACO
      074362 125252
14657 074364 040300 000000 000000 .WORD 40300,0,0,0 ;FSRC
      074372 000000
14658 074374 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
      074402 000000
14659 074404 076700 000000 000000 .WORD 76700,0,0,-1 ;INTEGER RESULT
      074412 177777
14660 074414 000200 .WORD 200 ; TEST FPS
14661 074416 000204 .WORD 204 ;RESULT FPS
14662 ;9/ROUND THROUGH FRACTION
14663 074420 005037 003030 CLR @FLAG ;NO INTERRUPT
14664 074424 004767 000424 JSR R7,MDDSUB ;DO TEST
14665 074430 041525 052525 052525 .WORD 41525,052525,52525,52525 ;ACO
      074436 052525
14666 074440 040300 000000 000000 .WORD 40300,0,0,0 ;FSRC
      074446 000000
14667 074450 040177 177777 177777 .WORD 40177,-1,-1,177740 ;FRACTIONAL RESULT
      074456 177777
14668 074460 041636 000000 000000 .WORD 41636,0,0,0 ;INTEGER RESULT
      074466 000000
14669 074470 007700 .WORD 7700 ; TEST FPS
14670 074472 007700 .WORD 7700 ;RESULT FPS
14671 ;/OVERFLOW, TRAPS ENABLED
14672 074474 012737 000001 003030 MOV #1,@FLAG ;INTERRUPT
14673 074502 004767 000346 JSR R7,MDDSUB ;DO TEST
14674 074506 177777 177777 177777 .WORD 1,-1,1,1 ;ACO
      074514 177777
14675 074516 040400 000000 000000 .WORD 40400,0,0,0 ;FSRC
      074524 000000
14676 074526 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT

```


FLOATING POINT TESTS

```

074534 000000
14677 074536 100177 177777 177777 .WORD 100177, 1, 1, -1 ;INTEGER RESULT
074544 177777
14678 074546 007700 .WORD 7700 ; TEST FPS
14679 074550 107706 .WORD 107706 ;RESULT FPS
14680 074552 000010 .WORD 10 ;FEC
14681 ;/INTEGER CHOPPED TO 56 BITS
14682 074554 005037 003030 CLR #0FLAG ;NO INTERRUPT
14683 074560 004767 000270 JSR R7,MDDSUB ;DO TEST
14684 074564 056700 000000 000000 .WORD 56700,0,0,-1 ;ACO
074572 177777
14685 074574 044440 177777 177777 .WORD 44440,-1,-1,-1 ;FSRC
074602 177777
14686 074604 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
074612 000000
14687 074614 063161 100000 000001 .WORD 63161,100000,1,40775 ;INTEGER RESULT
074622 040775
14688 074624 000200 .WORD 200 ; TEST FPS
14689 074626 000204 .WORD 204 ;RESULT FPS
14690 ;/OVERFLOW, TRAPS DISABLED
14691 074630 012737 000002 003030 MOV #2,#0FLAG ;NO INTERRUPT
14692 074636 004767 000212 JSR R7,MDDSUB ;DO TEST
14693 074642 066600 000000 000000 .WORD 66600,0,0,0 ;ACO
074650 000000
14694 074652 066600 000000 000000 .WORD 66600,0,0,0 ;FSRC
074660 000000
14695 074662 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
074670 000000
14696 074672 015200 000000 000000 .WORD 15200,0,0,0 ;INTEGER RESULT
074700 000000
14697 074702 047700 .WORD 47700 ; TEST FPS
14698 074704 147706 .WORD 147706 ;RESULT FPS
14699 074706 000010 .WORD 10 ;FEC
14700 ;/UNDERFLOW, TRAPS DISABLED
14701 074710 012737 000002 003030 MOV #2,#0FLAG ;NO INTERRUPT
14702 074716 004767 000132 JSR R7,MDDSUB ;DO TEST
14703 074722 100277 000001 000002 .WORD 100277,1,2,-1 ;ACO
074730 177777
14704 074732 100300 000001 000001 .WORD 100300,1,1,1 ;FSRC
074740 000001
14705 074742 000000 000000 000000 .WORD 0,0,0,0 ;FRACTIONAL RESULT
074750 000000
14706 074752 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
074760 000000
14707 074762 005200 .WORD 5200 ; TEST FPS
14708 074764 005204 .WORD 5204 ;RESULT FPS
14709 074766 000010 .WORD 10 ;FEC
14710 ;/UNDERFLOW TRAPS ENABLED, UV AS RESULT
14711 074770 012737 000001 003030 MOV #1,#0FLAG ;INTERRUPT
14712 074776 004767 000052 JSR R7,MDDSUB ;DO TEST
14713 075002 100277 000001 000002 .WORD 100277,1,2, 1 ;ACO
075010 177777
14714 075012 100300 000001 000001 .WORD 100300,1,1,1 ;FSRC
075020 000001
14715 075022 040417 040001 077403 .WORD 40417,40001,77403,0 ;FRACTIONAL RESULT
075030 000000
14716 075032 000000 000000 000000 .WORD 0,0,0,0 ;INTEGER RESULT
    
```

FLOATING POINT TESTS

```

075040 000000
14717 075042 002200          .WORD 2200          ; TEST FPS
14718 075044 102200          .WORD 102200         ; RESULT FPS
14719 075046 000012          .WORD 12             ; FEC
14720
14721 075050 000167 000300   ;          JMP      HOP15          ; JUMP OVER SUBROUTINE
14722   ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14723   ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14724   ;
14725   ;          ACO
14726   ;          FSRC
14727   ;          FRACTIONAL RESULT
14728   ;          INTEGER RESULT
14729   ;          FPS BEFORE EXECUTION
14730   ;          FPS AFTER EXECUTION
14731   ;          (FEC)
14732   ;
14733   ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14734   ; *X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14735   ;
14736 075054 012605          MDDSUB: MOV      (SP)+,R5          ; RETURN ADDRESS TO USE AS POINTER
14737 075056 012737 075144 000244  MOV      #50#,R0#FPVEC        ; REDIRECT TRAP VECTOR
14738 075064 012702 000200          MOV      #200,R2            ; SET TO DOUBLE MODE FOR LOAD
14739 075070 170102          LDFPS   R2                  ; LOAD FPS
14740 075072 172415          LDD     (R5),ACO            ; LOAD ACO WITH TEST DATA
14741 075074 012701 073650          MOV      #MODGAR,R1         ; LOAD KNOWN INTO AC1
14742 075100 172511          LDD     (R1),AC1            ;
14743 075102 010501          MOV      R5,R1              ; POINT TO FSRC DATA
14744 075104 062701 000010          ADD     #10,R1
14745 075110 016502 000040          MOV      40(R5),R2          ; GET TEST FPS
14746 075114 170102          LDFPS   R2                  ; LOAD TEST FPS
14747   ;
14748 075116 171411          MODD    (R1),ACO            ; *TEST INSTRUCTION
14749 075120 170011          1$: SETD                      ; WAIT FOR POSSIBLE FPA TRAP.
14750   ;
14751   ; INSTRUCTION DIDNT TRAP
14752   ;
14753 075122 032737 000001 003030          BIT     #1,#FLAG            ; VERIFY A NO TRAP CONDITION
14754 075130 001426          BEQ     2$                  ; BRANCH IF GOOD
14755 075132 004737 140132          CALL    #DET FPA            ; DETERMINE FLOATING POINT FAULT. $$$
14756 075136 104003          ERROR   +3                  ; FPP ERROR
14757   ; INSTRUCTION SHOULD HAVE TRAPPED
14758 075140 000167 000042          JMP     2$                  ; REJOIN CODE
14759   ;
14760   ; INSTRUCTION TRAPPED
14761   ;
14762 075144 032737 000001 003030          50$: BIT     #1,#FLAG            ; SEE IF EXPECTING A TRAP
14763 075152 001005          BNE     51$                  ; BRANCH IF EXPECTING A TRAP
14764 075154 004737 140132          CALL    #DET FPA            ; DETERMINE FLOATING POINT FAULT. $$$
14765 075160 104003          ERROR   +3                  ; FPP ERROR
14766   ; INSTRUCTION WASNT SUPPOSE TO TRAP
14767 075162 000167 000020          JMP     2$                  ; REJOIN CODE
14768 075166 012604          51$: MOV      (SP)+,R4          ; SEE IF PC = INSTRUCTION
14769 075170 005726          TST     (SP)+                ; CLEAN UP STACK
14770 075172 022704 075120          CMP     #1$,R4              ;
14771 075176 001403          BEQ     2$                  ; BRANCH IF GOOD COMPARE
14772 075200 004737 140132          CALL    #DET FPA            ; DETERMINE FLOATING POINT FAULT. $$$

```

FLOATING POINT TESTS

```

14773 075204 104003          ERROR +3          ;FPP ERROR
14774                                     ;PC WAS INCORRECT
14775
14776          ;
14777          ;COMMON CODE FOR TRAP AND NO TRAP
14778 075206 170203          ;
14779 075210 012702 000200 2$:      STFPS   R3          ;SAVE FPS
14780 075214 170102          MOV     #200,R2       ;SET FPP TO DOUBLE
14781 075216 012701 003142          LDFPS  R2
14782          ;SAVE FRACTIONAL RESULT      MOV     #RECDST,R1    ;POINT TO RECEIVED DATA TABLE
14783 075222 174011          STD     ACO,(R1)       ;SAVE ACO RESULT
14784 075224 026503 000042          CMP     42(R5),R3     ;VERIFY STATUS
14785 075230 001403          BEQ     3$           ;BRANCH IF GOOD
14786 075232 004737 140132          CALL   @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14787 075236 104003          ERROR   +3          ;FPP ERROR
14788                                     ;BAD FPS
14789 075240 010504          3$:      MOV     R5,R4          ;POINT TO EXPECTED DATA
14790 075242 062704 000020          ADD     #20,R4
14791 075246 004767 042626          4$:      JSR     R7,DATVER        ;VERIFY DATA
14792 075252 005767 105642          TST     COUNT
14793 075256 001403          BEQ     5$           ;BRANCH IF GOOD
14794 075260 004737 140132          CALL   @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14795 075264 104003          ERROR   +3          ;FPP ERROR
14796                                     ;BAD ACO
14797          ;SAVE INTEGER RESULT
14798          ;
14799 075266 174111          5$:      STD     AC1,(R1)       ;SAVE AC1 RESULT
14800 075270 010504          MOV     R5,R4          ;POINT TO EXPECTED
14801 075272 062704 000030          ADD     #30,R4
14802 075276 004767 042576          JSR     R7,DATVER        ;VERIFY DATA
14803 075302 005767 105612          TST     COUNT
14804 075306 001403          BEQ     6$           ;BRANCH IF GOOD
14805 075310 004737 140132          CALL   @DETFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14806 075314 104003          ERROR   +3          ;FPP ERROR
14807                                     ;BAD AC1
14808 075316 005737 003030          6$:      TST     @FLAG
14809 075322 001002          BNE     7$           ;SEE IF NEED TO CHECK FEC
14810 075324 000165 000044          JMP     44(R5)        ;BRANCH IF NEED TO CHECK
14811 075330 170301          7$:      STST    R1          ;RETURN FROM TEST
14812 075332 016504 000044          MOV     44(R5),R4     ;SAVE FEC
14813 075336 020401          CMP     R4,R1         ;GET FEC
14814 075340 001403          BEQ     8$           ;VERIFY FEC
14815 075342 004737 140132          CALL   @DETFPA       ;BRANCH IF GOOD
14816 075346 104003          ERROR   +3          ;DETERMINE FLOATING POINT FAULT. $$$
14817                                     ;FPP ERROR
14818 075350 000165 000046          8$:      JMP     46(R5)        ;BAD FEC
14819                                     ;RETURN FROM TEST
14820 075354          ;
14821          ;HOP15:
14822          ;
14823          ;-----
14824          ;TEST STCFD
14825          ;
14826          ;MSFD:
14827 075354          ;
14828          ;
14829          ;1/AC=0
14830 075354 004767 000170          JSR     R7,SFDSUB     ;DO TEST
14831 075360 000177 000000 000000 .WORD  0177,0,0,1    ;ACO
    
```

FLOATING POINT TESTS

```

14832 075366 000001
14832 075370 000000 000000 000000 .WORD 0,0,0,0 ;RESULT
      075376 000000
14833 075400 047557 .WORD 47557 ; TEST FPS
14834 075402 047544 .WORD 47544 ;RESULT FPS
14835 ;2/AC>0, TRUNCATE
14836 075404 004767 000140 JSR R7,SFDSUB ;DO TEST
14837 075410 077577 177777 177777 .WORD 77577,-1,-1,-1 ;ACO
      075416 177777
14838 075420 077577 177777 000000 .WORD 77577,-1,0,0 ;RESULT
      075426 000000
14839 075430 007540 .WORD 7540 ; TEST FPS
14840 075432 007540 .WORD 7540 ;RESULT FPS
14841 ;3/AC<0, ROUND
14842 075434 004767 000110 JSR R7,SFDSUB ;DO TEST
14843 075440 100377 177777 100000 .WORD 100377,-1,100000,0 ;ACO
      075446 000000
14844 075450 100377 177777 000000 .WORD 100377,-1,0,0 ;RESULT
      075456 000000
14845 075460 007517 .WORD 7517 ; TEST FPS
14846 075462 007510 .WORD 7510 ;RESULT FPS
14847 ;4/AC=-0
14848 075464 004767 000060 JSR R7,SFDSUB ;DO TEST
14849 075470 100000 000000 000000 .WORD 100000,0,0,0 ;ACO
      075476 000000
14850 075500 000000 000000 000000 .WORD 0,0,0,0 ;RESULT
      075506 000000
14851 075510 007757 .WORD 7757 ; TEST FPS
14852 075512 007744 .WORD 7744 ;RESULT FPS
14853 ;5/AC<0
14854 075514 004767 000030 JSR R7,SFDSUB ;DO TEST
14855 075520 125252 125252 125252 .WORD 125252,125252,125252,125252 ;ACO
      075526 125252
14856 075530 125252 125252 000000 .WORD 125252,125252,0,0 ;RESULT
      075536 000000
14857 075540 000000 .WORD 0 ; TEST FPS
14858 075542 000010 .WORD 10 ;RESULT FPS
14859 ;
14860 075544 000167 000120 JMP HOP16 ;GET OVER SUBROUTINE
14861 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
14862 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
14863 ;STCFD
14864 ; ACO
14865 ; RESULT
14866 ; FPS BEFORE EXECUTION
14867 ; FPS AFTER EXECUTION
14868 ;
14869 ;THERE CAN BE NO TRAPS WITH THE STCFD INSTRUCTION
14870 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
14871 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
14872 ;
SFDSUB: MOV (SP),R5 ; RETURN ADDRESS TO USE AS POINTER
14873 075550 012605 MOV #50,#@FPVEC ;REDIRECT TRAP VECTOR
14874 075552 012737 075656 000244 MOV #200,R2 ;SET TO DOUBLE MODE FOR LOAD
14875 075560 012702 000200 LDFPS R2 ;LOAD FPS
14876 075564 170102 LDD (R5),ACO ;LOAD ACO WITH TEST DATA
14877 075566 172415 MOV @RECDST,R1 ;POINT TO RESULT AREA
14878 075570 012701 003142

```

FLOATING POINT TESTS

```

14879 075574 016502 000020      MOV    20(R5),R2      ;GET TEST FPS
14880 075600 170102             LDFPS  R2            ;LOAD TEST FPS
14881                          ;
14882 075602 176011      40$:  STCFD  ACO,(R1) ;*TEST INSTRUCTION
14883                          ;
14884                          ;INSTRUCTION DIDNT TRAP
14885                          ;VERIFY STATUS
14886                          ;
14887 075604 170203      2$:  STFPS  R3            ;SAVE FPS
14888 075606 016502 000022      MOV    22(R5),R2      ;GET EXPECTED STATUS
14889 075612 020203      CMP    R2,R3         ;VERIFY STATUS
14890 075614 001403      BEQ    3$            ;BRANCH IF GOOD
14891 075616 004737 140132      CALL  @#DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14892 075622 104003      ERROR  +3           ;FPP ERROR
14893                          ;BAD FPS
14894 075624 010504      3$:  MOV    R5,R4         ;POINT TO EXPECTED DATA
14895 075626 062704 000010      ADD   #10,R4
14896 075632 004767 042242      4$:  JSR   R7,DATVER      ;VERIFY DATA
14897 075636 005767 105256      TST   COUNT
14898 075642 001403      BEQ    5$            ;BRANCH IF GOOD
14899 075644 004737 140132      CALL  @#DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14900 075650 104003      ERROR  +3           ;FPP ERROR
14901                          ;BAD ACO
14902 075652 000165 000024      5$:  JMP    24(R5)       ;RETURN FROM TEST
14903                          ;
14904                          ;INSTRUCTION TRAPPED
14905                          ;
14906 075656 004737 140132      50$: CALL  @#DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
14907 075662 104003      ERROR  +3           ;FPP ERROR
14908                          ;INSTRUCTION WASNT SUPPOSE TO TRAP
14909 075664 000165 000024      JMP    24(R5)       ;RETURN FROM TEST
14910                          ;
14911 075670      HOP16:
14914                          ;
14915                          ;-----
14916                          ;TEST STCDF
14917                          ;
14918 075670      MSDF:
14919                          ;
14920                          ;1/AC=0
14921 075670 005037 003030      CLR   @#FLAG        ;NO INTERRUPT
14922 075674 004767 000220      JSR   R7,SDFSUB     ;DO TEST
14923 075700 000177 000000 000000      .WORD 177.0,0,0     ;ACO
14924 075710 000000 000000      .WORD 0,0           ;RESULT
14925 075714 000200      .WORD 200           ; TEST FPS
14926 075716 000204      .WORD 204           ;RESULT FPS
14927                          ;2/AC=-0
14928 075720 005037 003030      CLR   @#FLAG        ;NO INTERRUPT
14929 075724 004767 000170      JSR   R7,SDFSUB     ;DO TEST
14930 075730 100000 000300 000200      .WORD 100000,300,200,100 ;ACO
14931 075740 000000 000000      .WORD 0,0           ;RESULT
14932 075744 007777      .WORD 7777          ; TEST FPS
14933 075746 007744      .WORD 7744          ;RESULT FPS
14934                          ;3/AC>0, TRUNCATE
14935 075750 005037 003030      CLR   @#FLAG        ;NO INTERRUPT
    
```

FLOATING POINT TESTS

```

14936 075754 004767 000140      JSR    R7,SDFSUB      ;DO TEST
14937 075760 055555 055555 177777 .WORD  55555,55555,-1,-1      ;ACO
      075766 177777
14938 075770 055555 055555      .WORD  55555,55555      ;RESULT
14939 075774 000240      .WORD  240              ; TEST FPS
14940 075776 000240      .WORD  240              ;RESULT FPS
14941      ;4/AC<0, ROUND TO UNDEFINED VARIABLE
14942 076000 012737 000001 003030 MOV    #1,#0FLAG      ;INTERRUPT
14943 076006 004767 000106      JSR    R7,SDFSUB      ;DO TEST
14944 076012 077777 177777 100000 .WORD  77777,-1,100000,0    ;ACO
      076020 000000
14945 076022 000000 000000      .WORD  0,0              ;RESULT
14946 076026 001200      .WORD  1200             ; TEST FPS
14947 076030 101206      .WORD  101206           ;RESULT FPS
14948      ;5/AC<0, ROUND
14949 076032 005037 003030 CLR    #0FLAG          ;NO INTERRUPT
14950 076036 004767 000056      JSR    R7,SDFSUB      ;DO TEST
14951 076042 125252 125252 125252 .WORD  125252,125252,125252,125252 ;ACO
      076050 125252
14952 076052 125252 125253      .WORD  125252,125253    ;RESULT
14953 076056 007700      .WORD  7700             ; TEST FPS
14954 076060 007710      .WORD  7710             ;RESULT FPS
14955      ;6/ROUND TO UV, TRAPS DISABLED
14956 076062 012737 000002 003030 MOV    #2,#0FLAG      ;INTERRUPT
14957 076070 004767 000024      JSR    R7,SDFSUB      ;DO TEST
14958 076074 077777 177777 177777 .WORD  77777,-1,-1,0      ;ACO
      076102 000000
14959 076104 000000 000000      .WORD  0,0              ;RESULT
14960 076110 006700      .WORD  6700             ; TEST FPS
14961 076112 006706      .WORD  6706             ;RESULT FPS
14962      ;
14963 076114 000167 000232      JMP    HOP17           ;GET OVER SUBROUTINE
14964      ;
14965      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14966      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14967      ;STCDF
14968      ;
14969      ;          ACO
14970      ;          RESULT
14971      ;          FPS BEFORE EXECUTION
14972      ;          FPS AFTER EXECUTION
14973      ;
14974      ;A TRAP CAN ONLY OCCUR IF ROUNDING CAUSES OVERFLOW
14975      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14976      ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14977 076120 012605      SDFSUB: MOV    (SP)+,R5      ; RETURN ADDRESS TO USE AS POINTER
14978 076122 012737 076202 000244 MOV    #50#,#0FPVEC    ;REDIRECT TRAP VECTOR
14979 076130 012702 000200      MOV    #200,R2         ;SET TO DOUBLE MODE FOR LOAD
14980 076134 170102      LDFPS  R2              ;LOAD FPS
14981 076136 172415      LDD    (R5),ACO        ;LOAD ACO WITH TEST DATA
14982 076140 012701 003142      MOV    #RECDST,R1      ;POINT TO RESULT AREA
14983 076144 016502 000014      MOV    14(R5),R2       ;GET TEST FPS
14984 076150 170102      LDFPS  R2              ;LOAD TEST FPS
14985      ;
14986 076152 176011      40$: STCDF  ACO,(R1)    ;*TEST INSTRUCTION
14987 076154 170327      1$: STST   (PC)+       ;WAIT FOR POSSIBLE FPA TRAP.
14988 076156 000000      .WORD  0               ;STORE STATUS HERE.

```

FLOATING POINT TESTS

```

14989
14990 ; INSTRUCTION DIDNT TRAP
14991 ;
14992 076160 032737 000001 003030 BIT #1,0#FLAG ;VERIFY A NO TRAP CONDITION
14993 076166 001426 BEQ 2# ;BRANCH IF GOOD
14994 076170 004737 140132 CALL 0#DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
14995 076174 104003 ERROR +3 ;FPP ERROR
14996 ;INSTRUCTION SHOULD HAVE TRAPPED
14997 076176 000167 000042 JMP 2# ;REJOIN CODE
14998 ;
14999 ; INSTRUCTION TRAPPED
15000 ;
15001 076202 032737 000001 003030 50#: BIT #1,0#FLAG ;SEE IF EXPECTING A TRAP
15002 076210 001005 BNE 51# ;BRANCH IF EXPECTING A TRAP
15003 076212 004737 140132 CALL 0#DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15004 076216 104003 ERROR +3 ;FPP ERROR
15005 ;INSTRUCTION WASNT SUPPOSE TO TRAP
15006 076220 000167 000020 JMP 2# ;REJOIN CODE
15007 076224 012604 51#: MOV (SP)+,R4 ;SEE IF PC = INSTRUCTION
15008 076226 005726 TST (SP)+ ;CLEAN UP STACK
15009 076230 022704 076154 CMP #1#,R4 ;
15010 076234 001403 BEQ 2# ;BRANCH IF GOOD COMPARE
15011 076236 004737 140132 CALL 0#DFTFPA ;DETERMINE FLOATING POINT FAULT. $$$
15012 076242 104003 ERROR +3 ;FPP ERROR
15013 ;PC WAS INCORRECT
15014 ;
15015 ;COMMON CODE FOR TRAP AND NO TRAP
15016 ;VERIFY STATUS
15017 ;
15018 076244 170203 2#: STFPS R3 ;SAVE FPS
15019 076246 016502 000016 MOV 16(R5),R2 ;GET EXPECTED STATUS
15020 076252 020203 CMP R2,R3 ;VERIFY STATUS
15021 076254 001403 BEQ 3# ;BRANCH IF GOOD
15022 076256 004737 140132 CALL 0#DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15023 076262 104003 ERROR +3 ;FPP ERROR
15024 ;BAD FPS
15025 076264 010504 3#: MOV R5,R4 ;POINT TO EXPECTED DATA
15026 076266 062704 000010 ADD #10,R4
15027 076272 004767 041564 4#: JSR R7,DATVFR ;VERIFY DATA
15028 076276 005767 104616 TST COUNT
15029 076302 001403 BEQ 5# ;BRANCH IF GOOD
15030 076304 004737 140132 CALL 0#DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15031 076310 104003 ERROR +3 ;FPP ERROR
15032 ;BAD ACO
15033 076312 005737 003030 5#: TST 0#FLAG ;SEE IF NEED TO CHECK FEC
15034 076316 001002 BNE 7# ;BRANCH IF NEED TO CHECK
15035 076320 000165 000020 JMP 20(R5) ;RETURN FROM TEST
15036 ;
15037 ;VERIFY FEC
15038 ;
15039 076324 012704 003122 7#: MOV #RECFEC,R4 ;POINT TO FEC AREA
15040 076330 170314 STST (R4) ;SAVE FEC
15041 076332 021427 000010 CMP (R4),#10 ;VERIFY FEC FOR OVERFLOW
15042 076336 001403 BEQ 8# ;BRANCH IF GOOD
15043 076340 004737 140132 CALL 0#DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15044 076344 104003 ERROR +3 ;FPP ERROR
15045 ;BAD FEC

```

FLOATING POINT TESTS

```

15046 076346 000165 000020      8$:      JMP      20(R5)          ;RETURN FROM TEST
15047                          ;
15048 076352                    HOP17:
15051                          ;
15052                          ;-----
15053                          ;TEST STCFD  USING ILLEGAL ACCUMULATOR
15054                          ;
15055 076352                    MSFDI:
15056                          ;
15057 076352 012701 040000      MOV      #40000,R1          ;DISABLE INTERRUPTS
15058 076356 170101            LDFPS   R1                  ;
15059 076360 176006            STCFD   ACO,AC6             ;*TEST ILLEGAL INSTRUCTION
15060 076362 170202            STFPS   R2                  ;SAVE STATUS
15061 076364 170303            STST    R3                  ;SAVE FEC
15062 076366 022702 140000     CMP      #140000,R2        ;VERIFY FER SET
15063 076372 001403            BEQ     1$                  ;BRANCH IF ERROR RECEIVED
15064 076374 004737 140132     CALL    @DETFPA           ;DETERMINE FLOATING POINT FAULT. $$$
15065 076400 104003            ERROR   +3                  ;FPP ERROR
15066                          ;FER BIT NOT SET ON ILLEGAL INST.
15067 076402 022703 000002     1$:      CMP      #2,R3          ;VERIFY FEC = FLOATING OPDOE ERROR
15068 076406 001403            BEQ     2$                  ;BRANCH IF GOOD
15069 076410 004737 140132     CALL    @DETFPA           ;DETERMINE FLOATING POINT FAULT. $$$
15070 076414 104003            ERROR   +3                  ;FPP ERROR
15071                          ;FEC INCORRECT
15072 076416                    2$:
15073                          ;
15076                          ;
15077                          ;-----
15078                          ;TEST CLRD
15079                          ;
15080 076416                    MCLRQ:
15081                          ;
15082 076416 012701 003764      MOV      #TAB47,R1         ;POINT TO DATA
15083 076422 012704 000200      MOV      #200,R4          ;SET FPP STATUS TO DOUBLE
15084 076426 170104            LDFPS   R4                  ;
15085 076430 172411            LDD     (R1),ACO           ;
15086 076432 012701 003142      MOV      #RECDST,R1       ;POINT TO DATA BUFFER
15087 076436 174011            STD     ACO,(R1)          ;STORE GARBAGE
15088 076440 170411            CLRD   (R1)               ;CLEAR DATA BUFFER
15089 076442 012704 003314      MOV      #TAB6,R4         ;VERIFY BUFFER =0
15090 076446 004767 041426     JSR     R7,DATVER         ;
15091 076452 005767 104442     TST     COUNT             ;
15092 076456 001403            BEQ     1$                  ;BRANCH I RECDST = 0
15093 076460 004737 140132     CALL    @DETFPA           ;DETERMINE FLOATING POINT FAULT. $$$
15094 076464 104003            ERROR   +3                  ;FPP ERROR
15095                          ;RECDST NOT CLEARED
15096 076466 170202                    1$:      STFPS   R2                  ;SAVE STATUS
15097 076470 020227 000204     CMP      R2,#204          ;VERIFY STATUS
15098 076474 001403            BEQ     2$                  ;BRANCH IF GOOD
15099 076476 004737 140132     CALL    @DETFPA           ;DETERMINE FLOATING POINT FAULT. $$$
15100 076502 104003            ERROR   +3                  ;FPP ERROR
15101                          ;BAD STATUS
15102 076504                    2$:
15105                          ;
15106                          ;-----
15107                          ;TEST CLRD, ILLEGAL ACCUMULATOR
15108                          ;

```


FLOATING POINT TESTS

```

15109 076504
15110
15111 076504 012704 040200
15112 076510 170104
15113 076512 170406
15114 076514 170203
15115 076516 170305
15116 076520 022703 140200
15117 076524 001403
15118 076526 004737 140132
15119 076532 104003
15120
15121 076534 022705 000002
15122 076540 001403
15123 076542 004737 140132
15124 076546 104003
15125
15126 076550
15129
15130
15131
15132
15133 076550
15134
15135 076550 012704 003162
15136 076554 012714 147757
15137 076560 012701 003132
15138 076564 012737 076650 000244
15139 076572 170114
15140 076574 170211
15141 076576 020427 003162
15142 076602 001403
15143 076604 004737 140132
15144 076610 104003
15145
15146 076612 020127 003132
15147 076616 001403
15148 076620 004737 140132
15149 076624 104003
15150
15151 076626 023727 003132 147757
15152 076634 001412
15153 076636 004737 140132
15154 076642 104003
15155
15156 076644 000167 000012
15157
15158
15159
15160 076650 012600
15161 076652 012605
15162 076654 004737 140132
15163 076660 104003
15164
15165 076662
15168
15169

MCLRI:
;
; MOV #40200,R4 ;DISABLE INTERRUPTS
; LDFPS R4 ;LOAD STATUS
; CLRD R6 ;*TEST INSTRUCTION WITH ILLEGAL ACC
; STFPS R3 ;SAVE STATUS
; STST R5 ;SAVE FEC
; CMP #140200,R3 ;VERIFY ERROR
; BEQ 1# ;BRANCH IF FER SET
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
; ERROR IN FPS
; 1#: CMP #2,R5 ;VERIFY FEC =2 OPCODE ERROR
; BEQ 2# ;BRANCH IF GOOD
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
; BAD FEC
; 2#:
;
;-----
;TEST LDFPS, STFPS MODE 1
;
;MLS1:
;
; MOV #TSTLOC,R4 ;POINT R4 TO RAM
; MOV #147757,(R4) ;SETUP EXPECTED STATUS
; MOV #RECST,R1 ;SET BUFFER FOR RECEIVED STATUS
; MOV #10#,@FPVEC ;SETUP TRAP VECTOR
; LDFPS (R4) ;*TEST INSTRUCTION
; STFPS (R1) ;*TEST INSTRUCTION
; CMP R4,#TSTLOC ;VERIFY R4
; BEQ 1# ;BRANCH IF GOOD
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
;
; 1#: CMP R1,#RECST ;VERIFY R1
; BEQ 2# ;BRANCH IF GOOD
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
; BAD R1
; 2#: CMP @RECST,#147757 ;VERIFY STATUS
; BEQ 3# ;BRANCH F GOOD
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
; BAD STATUS
; JMP 3# ;GET OVER TRAP
;
;UNEXPECTED TRAP
;
; 10#: MOV (SP)+,R0 ;SAVE PC
; MOV (SP)+,R5 ;SAVE PS
; CALL @DEFPA ;DETERMINE FLOATING POINT FAULT. $$$
; ERROR +3 ;FPP ERROR
; UNEXPECTED TRAP
; 3#:
;
;-----

```

FLOATING POINT TESTS

```

15170          ;TEST LDFPS, STFPS MODE 2
15171          ;
15172 076662    ;MLS2:
15173          ;
15174 076662 012704 003162      MOV    #TSTLOC,R4          ;POINT R4 TO RAM
15175 076666 012714 145557      MOV    #145557,(R4)       ;SETUP EXPECTED STATUS
15176 076672 012701 003132      MOV    #RECST,R1         ;SET BUFFER FOR RECEIVED STATUS
15177 076676 012737 076762 000244  MOV    #10#,#FPVEC       ;SETUP TRAP VECTOR
15178 076704 170124              LDFPS (R4)+              ;*TEST INSTRUCTION
15179 076706 170221              STFPS (R1)+              ;*TEST INSTRUCTION
15180 076710 020427 003164      CMP    R4,#TSTLOC+2      ;VERIFY R4
15181 076714 001403              BEQ    1#                ;BRANCH IF GOOD
15182 076716 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15183 076722 104003              ERROR  +3                ;FPP ERROR
15184          ;
15185 076724 020127 003134      1# :  CMP    R1,#RECST+2    ;VERIFY R1
15186 076730 001403              BEQ    2#                ;BRANCH IF GOOD
15187 076732 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15188 076736 104003              ERROR  +3                ;FPP ERROR
15189          ;BAD R1
15190 076740 023727 003132 145557 2# :  CMP    #RECST,#145557    ;VERIFY STATUS
15191 076746 001412              BEQ    3#                ;BRANCH F GOOD
15192 076750 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15193 076754 104003              ERROR  +3                ;FPP ERROR
15194          ;BAD STATUS
15195 076756 000157 000012      JMP    3#                ;GET OVER TRAP
15196          ;
15197          ;UNEXPECTED TRAP
15198          ;
15199 076762 012600              10# : MOV    (SP)+,R0          ;SAVE PC
15200 076764 012605              MOV    (SP)+,R5          ;SAVE PS
15201 076766 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15202 076772 104003              ERROR  +3                ;FPP ERROR
15203          ;UNEXPECTED TRAP
15204 076774              3# :
15207          ;
15208          ;-----
15209          ;TEST LDFPS, STFPS MODE 3
15210          ;
15211 076774    ;MLS3:
15212          ;
15213 076774 012704 003162      MOV    #TSTLOC,R4          ;POINT R4 TO RAM
15214 077000 012737 003166 003162  MOV    #TSTLOC+4,#TSTLOC ;TSTLOC= DEFERRED ADDRESS
15215 077006 012737 147501 003166  MOV    #147501,#TSTLOC+4 ;SETUP EXPECTED STATUS
15216 077014 012701 003172      MOV    #TSTLOC+10,R1      ;R1 POINTS TO TSTLOC+10
15217 077020 012737 003132 003172  MOV    #RECST,#TSTLOC+10 ;SET DEFERRED BUFFER FOR RECEIVED STATUS
15218 077026 012737 077112 000244  MOV    #10#,#FPVEC       ;SETUP TRAP VECTOR
15219 077034 170134              LDFPS #R4)+              ;*TEST INSTRUCTION
15220 077036 170231              STFPS #R1)+              ;*TEST INSTRUCTION
15221 077040 020427 003164      CMP    R4,#TSTLOC+2      ;VERIFY R4
15222 077044 001403              BEQ    1#                ;BRANCH IF GOOD
15223 077046 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15224 077052 104003              ERROR  +3                ;FPP ERROR
15225          ;
15226 077054 020127 003174      1# .  CMP    R1,#TSTLOC+12    ;VERIFY R1
15227 077060 001403              BEQ    2#                ;BRANCH IF GOOD
15228 077062 004737 140132      CALL  #DEFPA            ;DETERMINE FLOATING POINT FAULT. $$$

```


FLOATING POINT TESTS

```

15290
15291 077240
15292
15293 077240 012704 003164
15294 077244 012737 003166 003162
15295 077252 012737 147501 003166
15296 077260 012701 003174
15297 077264 012737 003132 003172
15298 077272 012737 077356 000244
15299 077300 170154
15300 077302 170251
15301 077304 020427 003162
15302 077310 001403
15303 077312 004737 140132
15304 077316 104003
15305
15306 077320 020127 003172 1#:
15307 077324 001403
15308 077326 004737 140132
15309 077332 104003
15310
15311 077334 023727 003132 147501 2#:
15312 077342 001412
15313 077344 004737 140132
15314 077350 104003
15315
15316 077352 000167 000012
15317
15318
15319
15320 077356 012600
15321 077360 012605
15322 077362 004737 140132
15323 077366 104003
15324
15325 077370
15328
15329
15330
15331
15332 077370
15333
15334 077370 012704 003162
15335 077374 012737 140001 003166
15336 077402 012701 003272
15337 077406 012737 077476 000244
15338 077414 170164 000004
15339 077420 170261 177700
15340 077424 020427 003162
15341 077430 001403
15342 077432 004737 140132
15343 077436 104003
15344
15345 077440 020127 003272 1#:
15346 077444 001403
15347 077446 004737 140132
15348 077452 104003

;
;MLS5:
;
MOV #TSTLOC+2,R4 ;POINT R4 TO RAM
MOV #TSTLOC+4,#TSTLOC ;TSTLOC= DEFERRED ADDRESS
MOV #147501,#TSTLOC+4 ;SETUP EXPECTED STATUS
MOV #TSTLOC+12,R1 ;R1 POINTS TO 412
MOV #RECST,#TSTLOC+10 ;SET DEFERRED BUFFER FOR RECEIVED STATUS
MOV #10#,#FPVEC ;SETUP TRAP VECTOR
LDFPS #-(R4) ;*TEST INSTRUCTION
STFPS #-(R1) ;*TEST INSTRUCTION
CMP R4,#TSTLOC ;VERIFY R4
BEQ 1# ;BRANCH IF GOOD
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;
;
1#: CMP R1,#TSTLOC+10 ;VERIFY R1
BEQ 2# ;BRANCH IF GOOD
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;BAD R1
;
2#: CMP #RECST,#147501 ;VERIFY STATUS
BEQ 3# ;BRANCH IF GOOD
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;BAD STATUS
JMP 3# ;GET OVER TRAP
;
;UNEXPECTED TRAP
;
10#: MOV (SP)+,R0 ;SAVE PC
MOV (SP)+,R5 ;SAVE PS
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;UNEXPECTED TRAP
;
3#:
;
;-----
;TEST LDFPS, STFPS MODE 6
;
;MLS6:
;
MOV #TSTLOC,R4 ;POINT R4 TO RAM
MOV #140001,#TSTLOC+4 ;SETUP EXPECTED STATUS
MOV #TSTLOC+110,R1 ;R1 WILL POINT TO TESTLOC+10
MOV #10#,#FPVEC ;SETUP TRAP VECTOR
LDFPS 4(R4) ;*TEST INSTRUCTION
STFPS -100(R1) ;*TEST INSTRUCTION
CMP R4,#TSTLOC ;VERIFY R4
BEQ 1# ;BRANCH IF GOOD
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR
;
;
1#: CMP R1,#TSTLOC+110 ;VERIFY R1
BEQ 2# ;BRANCH IF GOOD
CALL #DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
ERROR +3 ;FPP ERROR

```

FLOATING POINT TESTS

```

15349
15350 077454 023727 003172 140001 2$: CMP @TSTLOC+10,@140001 ;BAD R1 ;VERIFY STATUS
15351 077462 001412 BEQ 3$ ;BRANCH F GOOD
15352 077464 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15353 077470 104003 ERROR +3 ;FPP ERROR
15354 ;BAD STATUS
15355 077472 000167 000012 JMP 3$ ;GET OVER TRAP
15356 ;
15357 ;UNEXPECTED TRAP
15358 ;
15359 077476 012600 10$: MOV (SP)+,R0 ;SAVE PC
15360 077500 012605 MOV (SP)+,R5 ;SAVE PS
15361 077502 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15362 077506 104003 ERROR +3 ;FPP ERROR
15363 ;UNEXPECTED TRAP
15364 077510 3$:
15367 ;
15368 ;-----
15369 ;TEST LDFPS, STFPS MODE 7
15370 ;
15371 077510 MLS7:
15372 ;
15373 077510 012704 003262 MOV @TSTLOC+100,R4 ;POINT R4 TO RAM
15374 077514 012737 003166 003162 MOV @TSTLOC+4,@TSTLOC ;TSTLOC= DEFERRED ADDRESS
15375 077522 012737 145501 003166 MOV @145501,@TSTLOC+4 ;SETUP EXPECTED STATUS
15376 077530 012701 003072 MOV @TSTLOC-70,R1 ;R1 POINTS TO TSTLOC+10
15377 077534 012737 003172 003164 MOV @TSTLOC+10,@TSTLOC+2 ;
15378 077542 012737 077632 000244 MOV @10$,@FPVEC ;SETUP TRAP VECTOR
15379 077550 170174 177700 LDFPS @-100(R4) ;*TEST INSTRUCTION
15380 077554 170271 000072 STFPS @72(R1) ;*TEST INSTRUCTION
15381 077560 020427 003262 CMP R4,@TSTLOC+100 ;VERIFY R4
15382 077564 001403 BEQ 1$ ;BRANCH IF GOOD
15383 077566 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15384 077572 104003 ERROR +3 ;FPP ERROR
15385 ;
15386 077574 020127 003072 1$: CMP R1,@TSTLOC-70 ;VERIFY R1
15387 077600 001403 BEQ 2$ ;BRANCH IF GOOD
15388 077602 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15389 077606 104003 ERROR +3 ;FPP ERROR
15390 ;BAD R1
15391 077610 023727 003172 145501 2$: CMP @TSTLOC+10,@145501 ;VERIFY STATUS
15392 077616 001412 BEQ 3$ ;BRANCH F GOOD
15393 077620 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15394 077624 104003 ERROR +3 ;FPP ERROR
15395 ;BAD STATUS
15396 077626 000167 000012 JMP 3$ ;GET OVER TRAP
15397 ;
15398 ;UNEXPECTED TRAP
15399 ;
15400 077632 012600 10$: MOV (SP)+,R0 ;SAVE PC
15401 077634 012605 MOV (SP)+,R5 ;SAVE PS
15402 077636 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15403 077642 104003 ERROR +3 ;FPP ERROR
15404 ;UNEXPECTED TRAP
15405 077644 3$:
15408 ;
15409 ;-----

```

FLOATING POINT TESTS

```

15410 ;TEST LDCLD MODE 27
15411 ;
15412 077644 ;MLDC2:
15413 ;
15414 077644 005001 CLR R1 ;INIT R1
15415 077646 012704 007700 MOV #7700,R4 ;FPS=DOUBLE, LONG
15416 077652 170104 LDFPS R4 ;
15417 077654 012737 077714 000244 MOV #10#,R0 ;SETUP WILD TRAP
15418 077662 177027 LDCLD (R7)+,ACO ;*TEST INSTRUCTION
15419 077664 005201 INC R1 ;
15420 077666 005201 INC R1 ;
15421 077670 005201 INC R1 ;
15422 077672 005201 INC R1 ;
15423 077674 020127 000003 CMP R1,#3 ;VERIFY
15424 077700 001412 BEQ 1$ ;BRANCH IF GOOD
15425 077702 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15426 077706 104003 ERROR +3 ;FPP ERROR
15427 ;INSTRUCTION FAILED
15428 077710 000167 000012 JMP 1$ ;JUMP OVER WILD TRAP
15429 077714 012600 10$: MOV (SP)+,R0 ;SAVE PC
15430 077716 012605 MOV (SP)+,R1 ;SAVE PS
15431 077720 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15432 077724 104003 ERROR +3 ;FPP ERROR
15433 ;WILD TRAP ON INSTRUCTION
15434 077726 012704 003324 1$: MOV #TAB6A,R4 ;POINT TO EXPECTED DATA
15435 077732 012701 003142 MOV #RECDST,R1 ;POINT TO DATA BUFFER
15436 077736 174011 STD ACO,(R1) ;VERIFY DATA
15437 077740 004767 040134 JSR R7,DATVER ;
15438 077744 005767 103150 TST COUNT ;
15439 077750 001403 BEQ 2$ ;BRANCH IF GOOD DATA
15440 077752 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15441 077756 104003 ERROR +3 ;FPP ERROR
15442 ;BAD DATA
15443 077760 2$:
15444 ;
15445 ;-----
15446 ;TEST LDCIF, LDCLF
15447 ;
15448 ;
15449 ;
15450 077760 ;MLCF:
15451 ;
15452 ;1/INT=0
15453 077760 004767 000500 JSR R7,LCFSUB ;DO TEST
15454 077764 000000 000000 .WORD 0,0 ;FSRC
15455 077770 000000 000000 .WORD 0,0 ;RESULT
15456 077774 000000 .WORD 0 ;TEST FPS
15457 077776 000004 .WORD 4 ;RESULT FPS
15458 ;2/INT=0,-1
15459 100000 004767 000460 JSR R7,LCFSUB ;DO TEST
15460 100004 000000 177777 .WORD 0,-1 ;FSRC
15461 100010 000000 000000 .WORD 0,0 ;RESULT
15462 100014 007440 .WORD 7440 ;TEST FPS
15463 100016 007444 .WORD 7444 ;RESULT FPS
15464 ;3/LONG=0
15465 100020 004767 000440 JSR R7,LCFSUB ;DO TEST
15466 100024 000000 000000 .WORD 0,0 ;FSRC
15467 100030 000000 000000 .WORD 0,0 ;RESULT
15468 100034 000100 .WORD 100 ;TEST FPS

```

FLOATING POINT TESTS

15469	100036	000104		.WORD	104		;RESULT FPS
15470							
15471	100040	004767	000420	;4/INT=40000			
15472	100044	040000	000000	JSR	R7,LCFSUB		;DO TEST
15473	100050	043600	000000	.WORD	40000,0		;FSRC
15474	100054	000017		.WORD	43600,0		;RESULT
15475	100056	000000		.WORD	17		;TEST FPS
15476				.WORD	0		;RESULT FPS
15477	100060	004767	000400	;5/LONG=1			
15478	100064	000000	000001	JSR	R7,LCFSUB		;DO TEST
15479	100070	040200	000000	.WORD	0,1		;FSRC
15480	100074	000117		.WORD	40200,0		;RESULT
15481	100076	000100		.WORD	117		;TEST FPS
15482				.WORD	100		;RESULT FPS
15483	100100	004767	000360	;6/INT=PATTERN			
15484	100104	000252	025252	JSR	R7,LCFSUB		;DO TEST
15485	100110	042052	000000	.WORD	252,25252		;FSRC
15486	100114	000000		.WORD	42052,0		;RESULT
15487	100116	000000		.WORD	0		;TEST FPS
15488				.WORD	0		;RESULT FPS
15489	100120	004767	000340	;7/INT=-40000			
15490	100124	140000	000000	JSR	R7,LCFSUB		;DO TEST
15491	100130	143600	000000	.WORD	-40000,0		;FSRC
15492	100134	000007		.WORD	143600,0		;RESULT
15493	100136	000010		.WORD	7		;TEST FPS
15494				.WORD	10		;RESULT FPS
15495	100140	004767	000320	;8/INT=-1			
15496	100144	177777	000000	JSR	R7,LCFSUB		;DO TEST
15497	100150	140200	000000	.WORD	-1,0		;FSRC
15498	100154	000007		.WORD	140200,0		;RESULT
15499	100156	000010		.WORD	7		;TEST FPS
15500				.WORD	10		;RESULT FPS
15501	100160	004767	000300	;9/INT=PATTERN			
15502	100164	125252	125252	JSR	R7,LCFSUB		;DO TEST
15503	100170	143652	126000	.WORD	125252,125252		;FSRC
15504	100174	000007		.WORD	143652,126000		;RESULT
15505	100176	000010		.WORD	7		;TEST FPS
15506				.WORD	10		;RESULT FPS
15507	100200	004767	000260	;10/LONG=40000			
15508	100204	040000	000000	JSR	R7,LCFSUB		;DO TEST
15509	100210	047600	000000	.WORD	40000,0		;FSRC
15510	100214	000117		.WORD	47600,0		;RESULT
15511	100216	000100		.WORD	117		;TEST FPS
15512				.WORD	100		;RESULT FPS
15513	100220	004767	000240	;11/LONG=1			
15514	100224	000000	000001	JSR	R7,LCFSUB		;DO TEST
15515	100230	040200	000000	.WORD	0,1		;FSRC
15516	100234	007557		.WORD	40200,0		;RESULT
15517	100236	007540		.WORD	7557		;TEST FPS
15518				.WORD	7540		;RESULT FPS
15519	100240	004767	000220	;12/LONG=PATTERN			
15520	100244	000000	000252	JSR	R7,LCFSUB		;DO TEST
15521	100250	042052	000000	.WORD	0,252		;FSRC
15522	100254	007557		.WORD	42052,0		;RESULT
15523	100256	007540		.WORD	7557		;TEST FPS
15524				.WORD	7540		;RESULT FPS
15525	100260	004767	000200	;13/LONG = -40000			
				JSR	R7,LCFSUB		;DO TEST

FLOATING POINT TESTS

```

15583
15584 ;NO TRAP CAN OCCUR
15585 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15586 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15587
15588 100464 012602 LCFSUB: MOV (SP)+,R2 ; RETURN ADDRESS TO USE AS POINTER
15589 100466 012737 100574 000244 MOV #50$,@FPVEC ;REDIRECT TRAP VECTOR
15590 100474 012701 003142 MOV @RECDST,R1 ;POINT TO RESULT AREA
15591 100500 016200 000010 MOV 10(R2),R0 ;GET TEST FPS
15592 100504 170100 LDFPS R0 ;LOAD TEST FPS
15593 100506 010204 MOV R2,R4 ;POINT TO TEST DATA
15594
15595 100510 177014 40$: LDCIF (R4),ACO ;*TEST INSTRUCTION (ACCORDING TO MODE)
15596
15597 ;VERIFY STATUS
15598
15599 100512 170203 2$: STFPS R3 ;SAVE FPS
15600 100514 012700 000200 MOV #200,R0 ;SET FPP STATUS TO DOUBLE
15601 100520 170100 LDFPS R0 ;
15602 100522 174011 STD ACO,(R1) ;SAVE TEST RESULT INTO RECDST
15603 100524 016200 000012 MOV 12(R2),R0 ;GET EXPECTED STATUS
15604 100530 020003 CMP R0,R3 ;VERIFY STATUS
15605 100532 001403 BEQ 3$ ;BRANCH IF GOOD
15606 100534 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15607 100540 104003 ERROR +3 ;FPP ERROR
15608 ;BAD FPS
15609 100542 010204 3$: MOV R2,R4 ;POINT TO EXPECTED DATA
15610 100544 062704 000004 ADD #4,R4
15611 100550 004767 037306 4$: JSR R7,DATVFR ;VERIFY DATA
15612 100554 005767 102340 TST COUNT
15613 100560 001403 BEQ 5$ ;BRANCH IF GOOD
15614 100562 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15615 100566 104003 ERROR +3 ;FPP ERROR
15616 ;BAD ACO
15617 100570 000162 000014 5$: JMP 14(R2) ;RETURN FROM TEST
15618
15619 ;INSTRUCTION TRAPPED
15620
15621 100574 012600 50$: MOV (SP)+,R0 ;SAVE PC
15622 100576 012605 MOV (SP)+,R5 ;SAVE PS
15623 100600 004737 140132 CALL @DETFPA ;DETERMINE FLOATING POINT FAULT. $$$
15624 100604 104003 ERROR +3 ;FPP ERROR
15625 ;INSTRUCTION WASNT SUPPOSE TO TRAP
15626 100606 000167 177756 JMP 5$ ;CONTINUE
15627 100612
15630
15631 ;-----
15632 ;TEST LDCID, LDCLD
15633
15634 100612 MLCD:
15635
15636 ;1/LONG=0
15637 100612 004767 000264 JSR R7,LCDSUB ;DO TEST
15638 100616 000000 000000 .WORD 0,0 ;FSRC
15639 100622 000000 000000 000000 .WORD 0,0,0,0 ;RESULT
15640 100632 007313 .WORD 7313 ; TEST FPS

```

FLOATING POINT TESTS

```

15641 100634 007304 .WORD 7304 ;RESULT FPS
15642 ;2/INT=0
15643 100636 004767 000240 JSR R7,LCDSUB ;DO TEST
15644 100642 000000 000001 .WORD 0,1 ;FSRC
15645 100646 040200 000000 000000 .WORD 40200,0,0,0 ;RESULT
15646 100654 000000
15646 100656 007757 .WORD 7757 ; TEST FPS
15647 100660 007740 .WORD 7740 ;RESULT FPS
15648 ;3/INT=40000
15649 100662 004767 000214 JSR R7,LCDSUB ;DO TEST
15650 100666 040000 177777 .WORD 40000,-1 ;FSRC
15651 100672 043600 000000 000000 .WORD 43600 0,0,0 ;RESULT
15652 100700 000000
15652 100702 007617 .WORD 7617 ; TEST FPS
15653 100704 007600 .WORD 7600 ;RESULT FPS
15654 ;4/INT=-40000
15655 100706 004767 000170 JSR R7,LCDSUB ;DO TEST
15656 100712 140000 177777 .WORD -40000,-1 ;FSRC
15657 100716 143600 000000 000000 .WORD 143600,0,0,0 ;RESULT
15658 100724 000000
15658 100726 007600 .WORD 7600 ; TEST FPS
15659 100730 007610 .WORD 7610 ;RESULT FPS
15660 ;5/LONG=40000
15661 100732 004767 000144 JSR R7,LCDSUB ;DO TEST
15662 100736 040000 000000 .WORD 40000,0 ;FSRC
15663 100742 047600 000000 000000 .WORD 47600,0,0,0 ;RESULT
15664 100750 000000
15664 100752 007757 .WORD 7757 ; TEST FPS
15665 100754 007740 .WORD 7740 ;RESULT FPS
15666 ;6/LONG=1
15667 100756 004767 000120 JSR R7,LCDSUB ;DO TEST
15668 100762 000000 000001 .WORD 0,1 ;FSRC
15669 100766 040200 000000 000000 .WORD 40200,0,0,0 ;RESULT
15670 100774 000000
15670 100776 000300 .WORD 300 ; TEST FPS
15671 101000 000300 .WORD 300 ;RESULT FPS
15672 ;7/LONG=-2
15673 101002 004767 000074 JSR R7,LCDSUB ;DO TEST
15674 101006 177777 177776 .WORD -1,-2 ;FSRC
15675 101012 140400 000000 000000 .WORD 140400,0,0,0 ;RESULT
15676 101020 000000
15676 101022 007300 .WORD 7300 ; TEST FPS
15677 101024 007310 .WORD 7310 ;RESULT FPS
15678 ;8/INT=PATTERN
15679 101026 004767 000050 JSR R7,LCDSUB ;DO TEST
15680 101032 123456 176543 .WORD 123456,176543 ;FSRC
15681 101036 143661 122000 000000 .WORD 143661,122000,0,0 ;RESULT
15682 101044 000000
15682 101046 000200 .WORD 200 ; TEST FPS
15683 101050 000210 .WORD 210 ;RESULT FPS
15684 ;9/LONG=PATTERN
15685 101052 004767 000024 JSR R7,LCDSUB ;DO TEST
15686 101056 125252 125252 .WORD 125252,125252 ;FSRC
15687 101062 147652 125252 126000 .WORD 147652,125252,126000,0 ;RESULT
15688 101070 000000
15688 101072 000300 .WORD 300 ; TEST FPS
15689 101074 000310 .WORD 310 ;RESULT FPS
    
```

FLOATING POINT TESTS

```

15690
15691 101076 000167 000126      ;      JMP      HOP19          ;GET OVER SUBROUTINE
15692
15693 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
15694 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
15695 ;LDCID, LDCLO
15696
15697 ;
15698 ;      FSRC
15699 ;      RESULT
15700 ;      FPS BEFORE EXECUTION
15701 ;      FPS AFTER EXECUTION
15702
15703 ;NO TRAP CAN OCCUR
15704 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
15705 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*
15706 101102 012602      LCDSUB: MOV      (SP)+,R2          ; RETURN ADDRESS TO USE AS POINTER
15707 101104 012737 101212 000244  MOV      #50$,#FPVEC          ;REDIRECT TRAP VECTOR
15708 101112 012701 003142      MOV      #RECDST,R1          ;POINT TO RESULT AREA
15709 101116 016200 000014      MOV      14(R2),R0          ;GET TEST FPS
15710 101122 170100      LDFPS   R0                  ;LOAD TEST FPS
15711 101124 010204      MOV      R2,R4              ;POINT TO TEST DATA
15712
15713 101126 177014      40$:   LDCID   (R4),ACO          ;*TEST INSTRUCTION (ACCORDING TO MODE)
15714
15715 ;VERIFY STATUS
15716
15717 101130 170203      2$:   STFPS   R3              ;SAVE FPS
15718 101132 012700 000200      MOV      #200,R0           ;SET FPP STATUS TO DOUBLE
15719 101136 170100      LDFPS   R0
15720 101140 174011      STD     ACO,(R1)           ;SAVE TEST RESULT INTO RECDST
15721 101142 016200 000016      MOV      16(R2),R0          ;GET EXPECTED STATUS
15722 101146 020003      CMP     R0,R3              ;VERIFY STATUS
15723 101150 001403      BEQ     3$                 ;BRANCH IF GOOD
15724 101152 004737 140132      CALL   #DETFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15725 101156 104003      ERROR  +3                 ;FPP ERROR
15726 ;BAD FPS
15727 101160 010204      3$:   MOV     R2,R4           ;POINT TO EXPECTED DATA
15728 101162 062704 000004      ADD     #4,R4
15729 101166 004767 036706      4$:   JSR     R7,DATVER          ;VERIFY DATA
15730 101172 005767 101722      TST    COUNT
15731 101176 001403      BEQ     5$                 ;BRANCH IF GOOD
15732 101200 004737 140132      CALL   #DETFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15733 101204 104003      ERROR  +3                 ;FPP ERROR
15734 ;BAD ACO
15735 101206 000162 000020      5$:   JMP     20(R2)          ;RETURN FROM TEST
15736
15737 ;INSTRUCTION TRAPPED
15738
15739 101212 012600      50$:  MOV     (SP)+,R0          ;SAVE PC
15740 101214 012605      MOV     (SP)+,R5          ;SAVE PS
15741 101216 004737 140132      CALL   #DETFPA            ;DETERMINE FLOATING POINT FAULT. $$$
15742 101222 104003      ERROR  +3                 ;FPP ERROR
15743 ;INSTRUCTION WASNT SUPPOSE TO TRAP
15744 101224 000167 177756      JMP     5$                 ;CONTINUE
15745
15746 101230      HOP19:

```

FLOATING POINT TESTS

```

15749 ;
15750 ;-----
15751 ;TEST LDEXP
15752 ;DOUBLE
15753 ;
15754 101230 MLXP:
15755 ;
15756 ;1/EXP=10 AC=NEG
15757 101230 005037 003030 CLR @FLAG ;NO INTERRUPTS
15758 101234 004767 001140 JSR R7,LXPSUB ;DO TEST
15759 101240 123456 067012 025252 .WORD 123456,67012,25252,171717 ;ACO
15760 101250 000010 .WORD 10 ;EXP
15761 101252 142056 067012 025252 .WORD 142056,67012,25252,171717 ;RESULT
15762 101262 007757 .WORD 7757 ; TEST FPS
15763 101264 007750 .WORD 7750 ;RESULT FPS
15764 ;2/EXP=177 - ACO=POS
15765 101266 005037 003030 CLR @FLAG ;NO INTERRUPTS
15766 101272 004767 001102 JSR R7,LXPSUB ;DO TEST
15767 101276 023456 070123 100000 .WORD 23456,70123,100000,1 ;ACO
15768 101306 000177 .WORD 177 ;EXP
15769 101310 077656 070123 100000 .WORD 77656,70123,100000,1 ;RESULT
15770 101320 007700 .WORD 7700 ; TEST FPS
15771 101322 007700 .WORD 7700 ;RESULT FPS
15772 ;3/EXP=56
15773 101324 005037 003030 CLR @FLAG ;NO INTERRUPTS
15774 101330 004767 001044 JSR R7,LXPSUB ;DO TEST
15775 101334 055555 044444 033333 .WORD 55555,44444,33333,22222 ;ACO
15776 101344 000056 .WORD 56 ;EXP
15777 101346 053555 044444 033333 .WORD 53555,44444,33333,22222 ;RESULT
15778 101356 007757 .WORD 7757 ; TEST FPS
15779 101360 007740 .WORD 7740 ;RESULT FPS
15780 ;4/EXP=-151, ACO=UV
15781 101362 005037 003030 CLR @FLAG ;NO INTERRUPTS
15782 101366 004767 001006 JSR R7,LXPSUB ;DO TEST
15783 101372 100077 177777 177777 .WORD 100077,-1,-1,-2 ;ACO
15784 101402 177623 .WORD -155 ;EXP
15785 101404 104677 177777 177777 .WORD 104677,-1,-1,-2 ;RESULT
15786 101414 007757 .WORD 7757 ; TEST FPS
15787 101416 007750 .WORD 7750 ;RESULT FPS
15788 ;5/EXP=-177
15789 101420 005037 003030 CLR @FLAG ;NO INTERRUPTS
15790 101424 004767 000750 JSR R7,LXPSUB ;DO TEST
15791 101430 000177 177777 177777 .WORD 177,-1,-1,-2 ;ACO
15792 101440 177601 .WORD 177 ;EXP
15793 101442 000377 177777 177777 .WORD 377,-1,-1,2 ;RESULT
15794 101452 007700 .WORD 7700 ; TEST FPS
15795 101454 007700 .WORD 7700 ;RESULT FPS

```

FLOATING POINT TESTS

```

15796                                     ;6/EXP=-200, UNDERFLOW
15797 101456 012737 000001 003030      MOV    #1,0#FLAG                ; INTERRUPTS
15798 101464 004767 000710              JSR    R7,LXPSUB                ;DO TEST
15799 101470 030131 032334 035363      .WORD  30131,32334,35363,73031  ;ACO
      101476 073031
15800 101500 177600                      .WORD  -200                    ;EXP
15801 101502 000131 032334 035363      .WORD  131,32334,35363,73031  ;RESULT
      101510 073031
15802 101512 007740                      .WORD  7740                    ; TEST FPS
15803 101514 107744                      .WORD  107744                 ;RESULT FPS
15804 101516 000012                      .WORD  12                      ;FEC
15805
15806 101520 012737 000001 003030      ;7/EXP=LARGEST NEGATIVE
15807 101526 004767 000646              MOV    #1,0#FLAG                ;EXPECT INTERRUPTS
15808 101532 000000 000123 000456      JSR    R7,LXPSUB                ;DO TEST
      101540 000001                      .WORD  0,123,456,1            ;ACO
15809 101542 100000                      .WORD  100000                 ;EXP
15810 101544 040000 000123 000456      .WORD  40000,123,456,1        ;RESULT
      101552 000001
15811 101554 002200                      .WORD  2200                   ; TEST FPS
15812 101556 102200                      .WORD  102200                 ;RESULT FPS
15813 101560 000012                      .WORD  12                      ;FEC
15814
15815 101562 012737 000001 003030      ;8/EXP=-200, NEG. ACO
15816 101570 004767 000604              MOV    #1,0#FLAG                ; INTERRUPTS
15817 101574 111111 100000 100000      JSR    R7,LXPSUB                ;DO TEST
      101602 177777                      .WORD  111111,100000,100000,-1 ;ACO
15818 101604 177600                      .WORD  -200                   ;EXP
15819 101606 100111 100000 100000      .WORD  100111,100000,100000,-1 ;RESULT
      101614 177777
15820 101616 002217                      .WORD  2217                   ; TEST FPS
15821 101620 102214                      .WORD  102214                 ;RESULT FPS
15822 101622 000012                      .WORD  12                      ;FEC
15823
15824 101624 012737 000002 003030      ;9/EXP=-1743, FIU=0
15825 101632 004767 000542              MOV    #2,0#FLAG                ;NO INTERRUPTS
15826 101636 123456 012346 012346      JSR    R7,LXPSUB                ;DO TEST
      101644 000123                      .WORD  123456,12346,12346,123  ;ACO
15827 101646 176035                      .WORD  -1743                  ;EXP
15828 101650 000000 000000 000000      .WORD  0,0,0,0                ;RESULT
      101656 000000
15829 101660 005700                      .WORD  5700                   ; TEST FPS
15830 101662 005704                      .WORD  5704                   ;RESULT FPS
15831 101664 000012                      .WORD  12                      ;FEC
15832
15833 101666 012737 000002 003030      ;10/EXP = -16616, FID=1
15834 101674 004767 000500              MOV    #2,0#FLAG                ;NO INTERRUPTS
15835 101700 000377 123456 065432      JSR    R7,LXPSUB                ;DO TEST
      101706 000001                      .WORD  377,123456,65432,1      ;ACO
15836 101710 161162                      .WORD  -16616                 ;EXP
15837 101712 074577 123456 065432      .WORD  74577,123456,65432,1  ;RESULT
      101720 000001
15838 101722 047700                      .WORD  47700                  ; TEST FPS
15839 101724 147700                      .WORD  147700                 ;RESULT FPS
15840 101726 000012                      .WORD  12                      ;FEC
15841
15842 101730 005037 003030      ;11/EXP=177, ACO=UNDEFINED VARIABLE
      CLR    0#FLAG                ;NO INTERRUPTS

```

FLOATING POINT TESTS

```

15843 101734 004767 000440          JSR    R7,LXPSUB          ;DO TEST
15844 101740 100177 177777 177777 .WORD  100177,-1,-1,-1    ;ACO
      101746 177777
15845 101750 000177                .WORD  177              ;EXP
15846 101752 177777 177777 177777 .WORD  1,-1,-1, 1        ;RESULT
      101760 177777
15847 101762 007700                .WORD  7700             ; TEST FPS
15848 101764 007710                .WORD  7710             ;RESULT FPS
15849                                ;12/EXP=150 ACO=POS
15850 101766 005037 003030          CLR    #0,0#FLAG        ;NO INTERRUPT
15851 101772 004767 000402          JSR    R7,LXPSUB          ;DO TEST
15852 101776 000200 000100 000200 .WORD  200,100,200,300   ;ACO
      102004 000300
15853 102006 000150                .WORD  150              ;EXP
15854 102010 072000 000100 000200 .WORD  72000,100,200,300 ;RESULT
      102016 000300
15855 102020 007717                .WORD  7717             ; TEST FPS
15856 102022 007700                .WORD  7700             ;RESULT FPS
15857                                ;13/EXP=200, ACO=NEG
15858 102024 012737 000001 003030  MOV    #1,0#FLAG
15859 102032 004767 000342          JSR    R7,LXPSUB          ;DO TEST
15860 102036 177777 177777 177777 .WORD  -1,-1,-1,-1      ;ACO
      102044 177777
15861 102046 000200                .WORD  200              ;EXP
15862 102050 100177 177777 177777 .WORD  100177,-1,-1,-1   ;RESULT
      102056 177777
15863 102060 007705                .WORD  7705             ; TEST FPS
15864 102062 107716                .WORD  107716           ;RESULT FPS
15865 102064 000010                .WORD  10               ;FEC
15866                                ;14/EXP=400, FID
15867 102066 012737 000002 003030  MOV    #2,0#FLAG        ;INTERRUPT
15868 102074 004767 000300          JSR    R7,LXPSUB          ;DO TEST
15869 102100 000555 177777 177776 .WORD  555,-1,-2,-3     ;ACO
      102106 177775
15870 102110 000100                .WORD  400              ;EXP
15871 102112 040155 177777 177776 .WORD  40155,-1,-2, 3    ;RESULT
      102120 177775
15872 102122 047700                .WORD  47700            ; TEST FPS
15873 102124 147702                .WORD  147702           ;RESULT FPS
15874 102126 000010                .WORD  10               ;FEC
15875                                ;15/EXP=11011 FIU=0
15876 102130 012737 000000 003030  MOV    #0,0#FLAG        ;NO INTERRUPT
15877 102136 004767 000236          JSR    R7,LXPSUB          ;DO TEST
15878 102142 177773 177777 177776 .WORD  177773,-1, 2,-3   ;ACO
      102150 177775
15879 102152 011011                .WORD  11011            ;EXP
15880 102154 000000 000000 000000 .WORD  0,0,0,0           ;RESULT
      102162 000000
15881 102164 006700                .WORD  6700             ; TEST FPS
15882 102166 006706                .WORD  6706             ;RESULT FPS
15883                                ;16/EXP=LARGEST POSITIVE
15884 102170 012737 000001 003030  MOV    #1,0#FLAG        ;INTERRUPT
15885 102176 004767 000176          JSR    R7,LXPSUB          ;DO TEST
15886 102202 123456 000100 000100 .WORD  123456,100,100,200 ;ACO
      102210 000200
15887 102212 077777                .WORD  77777            ;EXP
15888 102214 137656 000100 000100 .WORD  137656,100,100,200 ;RESULT

```

FLOATING POINT TESTS

```

15889 102224 007740           .WORD 7740           ; TEST FPS
15890 102226 107752           .WORD 107752        ; RESULT FPS
15891 102230 000010           .WORD 10           ; FEC
15892      ;17/FLOATING
15893 102232 005037 003030     CLR  #0#FLAG        ; NO INTERRUPT
15894 102236 004767 000136     JSR  R7,LXPSUB      ; DO TEST
15895 102242 123456 023465 000555 .WORD 123456,23465,555,444 ; ACO
      102250 000444
15896 102252 000050           .WORD 50           ; EXP
15897 102254 152056 023465 000555 .WORD 152056,23465,555,444 ; RESULT
      102262 000444
15898 102264 007500           .WORD 7500         ; TEST FPS
15899 102266 007510           .WORD 7510         ; RESULT FPS
15900      ;18/FLOATING UNDERFLOW
15901 102270 012737 000001 003030     MOV  #1,#0#FLAG    ; INTERRUPT
15902 102276 004767 000076     JSR  R7,LXPSUB      ; DO TEST
15903 102302 000333 000444 000555 .WORD 333,444,555,666 ; ACO
      102310 000666
15904 102312 177600           .WORD -200         ; EXP
15905 102314 000133 000444 000555 .WORD 133,444,555,666 ; RESULT
      102322 000666
15906 102324 007500           .WORD 7500         ; TEST FPS
15907 102326 107504           .WORD 107504       ; RESULT FPS
15908 102330 000012           .WORD 12           ; FEC
15909      ;19/FLOATING OVERFLOW
15910 102332 012737 000001 003030     MOV  #1,#0#FLAG    ; INTERRUPT
15911 102340 004767 000034     JSR  R7,LXPSUB      ; DO TEST
15912 102344 012346 000123 000345 .WORD 12346,123,345,456 ; ACO
      102352 000456
15913 102354 000400           .WORD 400          ; EXP
15914 102356 040146 000123 000345 .WORD 40146,123,345,456 ; RESULT
      102364 000456
15915 102366 007400           .WORD 7400         ; TEST FPS
15916 102370 107402           .WORD 107402       ; RESULT FPS
15917 102372 000010           .WORD 10           ; FEC
15918      ;
15919 102374 000167 000250     JMP  HOP20          ; GET OVER SUBROUTINE
15920      ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15921      ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15922      ;LDEXP
15923      ;
15924      ; ACO
15925      ; EXPONENT
15926      ; RESULT
15927      ; FPS BEFORE EXECUTION
15928      ; FPS AFTER EXECUTION
15929      ; (FEC)
15930      ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15931      ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15932      ;
15933 102400 012602           LXPSUB: MOV (SP)+,R2 ; RETURN ADDRESS TO USE AS POINTER
15934 102402 012737 102470 000244     MOV  #50#,#0#FPVEC ; REDIRECT TRAP VECTOR
15935 102410 012701 003142     MOV  #RECDST,R1    ; POINT TO RESULT AREA
15936 102414 012700 000200     MOV  #200,R0       ; SET FPS TO DOUBLE
15937 102420 170100           LDFPS R0           ;
15938 102422 010204           MOV  R2,R4         ; POINT TO ACO DATA
    
```

FLOATING POINT TESTS

```

15939 102424 172414          LDD      (R4),ACO          ;LOAD ACO
15940 102426 016200 000022  MOV      22(R2),R0        ;GET TEST FPS
15941 102432 170100          LDFPS   R0                ;LOAD TEST FPS
15942 102434 016204 000010  MOV      10(R2),R4        ;POINT TO TEST DATA
15943                               ;
15944 102440 176404          40$:    LDEXP   R4,ACO        ;*TEST INSTRUCTION (ACCORDING TO MODE)
15945 102442 170327          1$:    STST    (PC)+        ;WAIT FOR POSSIBLE FPA TRAP.
15946 102444 000000          .WORD   0                ;STORE STATUS HERE
15947                               ;
15948                               ;INSTRUCTION DIDNT TRAP
15949                               ;
15950 102446 032737 000001 003030  BIT      #1,#FLAG        ;VERIFY A NO TRAP CONDITION
15951 102454 001426          BEQ      2$                ;BRANCH IF GOOD
15952 102456 004737 140132  CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
15953 102462 104003          ERROR   +3              ;FPP ERROR
15954                               ;INSTRUCTION SHOULD HAVE TRAPPED
15955 102464 000167 000042  JMP      2$                ;REJOIN CODE
15956                               ;
15957                               ;INSTRUCTION TRAPPED
15958                               ;
15959 102470 032737 000001 003030 50$:    BIT      #1,#FLAG        ;SEE IF EXPECTING A TRAP
15960 102476 001005          BNE      51$              ;BRANCH IF EXPECTING A TRAP
15961 102500 004737 140132  CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
15962 102504 104003          ERROR   +3              ;FPP ERROR
15963                               ;INSTRUCTION WASNT SUPPOSE TO TRAP
15964 102506 000167 000020  JMP      2$                ;REJOIN CODE
15965 102512 012604          51$:    MOV      (SP)+,R4        ;SEE IF PC = INSTRUCTION
15966 102514 005726          TST     (SP)+            ;CLEAN UP STACK
15967 102516 022704 102442  CMP      #1$,R4          ;
15968 102522 001403          BEQ      2$                ;BRANCH IF GOOD COMPARE
15969 102524 004737 140132  CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
15970 102530 104003          ERROR   +3              ;FPP ERROR
15971                               ;PC WAS INCORRECT
15972                               ;
15973                               ;COMMON CODE FOR TRAP AND NO TRAP
15974                               ;VERIFY STATUS
15975                               ;
15976 102532 170203          2$:    STFPS   R3                ;SAVE FPS
15977 102534 012700 000200  MOV      #200,R0        ;SETUP FPS
15978 102540 170100          LDFPS   R0                ;FPS=200
15979 102542 174011          STD     ACO,(R1)        ;GET RESULT
15980 102544 016200 000024  MOV      24(R2),R0        ;GET EXPECTED STATUS
15981 102550 020003          CMP     R0,R3            ;VERIFY STATUS
15982 102552 001403          BEQ     3$                ;BRANCH IF GOOD
15983 102554 004737 140132  CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
15984 102560 104003          ERROR   +3              ;FPP ERROR
15985                               ;BAD FPS
15986 102562 010204          3$:    MOV      R2,R4                ;POINT TO EXPECTED DATA
15987 102564 062704 000012  ADD     #12,R4          ;
15988 102570 004767 035304  4$:    JSR     R7,DATVER        ;VERIFY DATA
15989 102574 005767 100320  TST     COUNT            ;
15990 102600 001403          BEQ     5$                ;BRANCH IF GOOD
15991 102602 004737 140132  CALL     @DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
15992 102606 104003          ERROR   +3              ;FPP ERROR
15993                               ;BAD ACO
15994 102610 005737 003030  5$:    TST     @FLAG            ;SEE IF NEED TO CHECK FEC
15995 102614 001002          BNE     7$                ;BRANCH IF NEED TO CHECK

```


FLOATING POINT TESTS

```

15996 102616 000162 000026          JMP      26(R2)          ;RETURN FROM TEST
15997          ;VERIFY FEC
15998 102622 012704 003122          7#: MOV     @RECFEC,R4    ;POINT TO FEC AREA
15999 102626 170314          STST    (R4)           ;SAVE FEC
16000 102630 021462 000026          CMP     (R4),26(R2)    ;VERIFY FEC FOR OVERFLOW
16001 102634 001403          BEQ     8#             ;BRANCH IF GOOD
16002 102636 004737 140132          CALL   @DETFPA        ;DETERMINE FLOATING POINT FAULT. ###
16003 102642 104003          ERROR   +3           ;FPP ERROR
16004          ;BAD FEC
16005 102644 000162 000030          8#: JMP     30(R2)      ;RETURN FROM TEST
16006          ;
16007 102650          ;HOP20:
16010          ;
16011          ;-----
16012          ;TEST STCDI, STCDL
16013          ;
16014 102650          ;MSCD:
16015          ;
16016          ;1/ACO=0, INT
16017 102650 005037 003030          CLR     @FLAG          ;NO INTERRUPTS
16018 102654 004767 000610          JSR     R7,SCDSUB      ;DO TEST
16019 102660 000177 000000 000000          .WORD  0177,0,0,0     ;ACO
16020          .WORD  0,-1          ;RESULT
16021 102674 007640          .WORD  7640           ;TEST FPS
16022 102676 007644          .WORD  7644           ;RESULT FPS
16023          ;2/ACO=-0, LONG
16024 102700 005037 003030          CLR     @FLAG          ;INTERRUPT
16025 102704 004767 000560          JSR     R7,SCDSUB      ;DO TEST
16026 102710 100177 177777 177777          .WORD  100177,-1,-1,-1 ;ACO
16027          .WORD  0,0           ;RESULT
16028 102724 007700          .WORD  7700           ;TEST FPS
16029 102726 007704          .WORD  7704           ;RESULT FPS
16030          ;3/EXP=100, LONG
16031 102730 005037 003030          CLR     @FLAG          ;NO INTERRUPT
16032 102734 004767 000530          JSR     R7,SCDSUB      ;DO TEST
16033 102740 020000 000000 000000          .WORD  20000,0,0,0    ;ACO
16034          .WORD  0,0           ;RESULT
16035 102754 000300          .WORD  300            ;TEST FPS
16036 102756 000304          .WORD  304            ;RESULT FPS
16037          ;4/EXP=200, BAISED 0, INT, ROUND
16038 102760 005037 003030          CLR     @FLAG          ;INTERRUPT
16039 102764 004767 000500          JSR     R7,SCDSUB      ;DO TEST
16040 102770 140177 177777 000001          .WORD  140177,177777,1,1 ;ACO
16041          .WORD  0,0           ;RESULT
16042 103004 007700          .WORD  7700           ;TEST FPS
16043 103006 007704          .WORD  7704           ;RESULT FPS
16044          ;5/LONG
16045 103010 005037 003030          CLR     @FLAG          ;INTERRUPT
16046 103014 004767 000450          JSR     R7,SCDSUB      ;DO TEST
16047 103020 047667 075757 157737          .WORD  47667,75757,157737,167773 ;ACO
16048          .WORD  55675,173757          ;RESULT
16049 103034 007717          .WORD  7717           ;TEST FPS

```

FLOATING POINT TESTS

16050	103036	007700			.WORD 7700	;RESULT FPS
16051					:6/LONG, EXP=2**32	
16052	103040	005037	003030		CLR @FLAG	;NO INTERRUPT
16053	103044	004767	000420		JSR R7,SCDSUB	;DO TEST
16054	103050	046400	000000	000000	.WORD 46400,0,0,0	;ACO
	103056	000000				
16055	103060	001000	000000		.WORD 1000,0	;RESULT
16056	103064	007700			.WORD 7700	; TEST FPS
16057	103066	007700			.WORD 7700	;RESULT FPS
16058					:7/LONG, EXP>2**32	
16059	103070	012737	000001	003030	MOV @1,@FLAG	;INTERRUPT
16060	103076	004767	000366		JSR R7,SCDSUB	;DO TEST
16061	103102	077607	000000	000000	.WORD 77607,0,0,0	;ACO
	103110	000000				
16062	103112	000000	000000		.WORD 0,0	;RESULT
16063	103116	007700			.WORD 7700	; TEST FPS
16064	103120	107705			.WORD 107705	;RESULT FPS
16065					:8/INT, EXP=2**15	
16066	103122	005037	003030		CLR @FLAG	;NO INTERRUPTS
16067	103126	004767	000336		JSR R7,SCDSUB	;DO TEST
16068	103132	043200	000000	000000	.WORD 43200,0,0,0	;ACO
	103140	000000				
16069	103142	010000	177777		.WORD 10000,-1	;RESULT
16070	103146	007600			.WORD 7600	; TEST FPS
16071	103150	007600			.WORD 7600	;RESULT FPS
16072					:9/INT, EXP>2**15	
16073	103152	012737	000001	003030	MOV @1,@FLAG	;INTERRUPT
16074	103160	004767	000304		JSR R7,SCDSUB	;DO TEST
16075	103164	077777	177777	177777	.WORD 77777,-1,-1,-1	;ACO
	103172	177777				
16076	103174	000000	177777		.WORD 0,-1	;RESULT
16077	103200	007600			.WORD 7600	; TEST FPS
16078	103202	107605			.WORD 107605	;RESULT FPS
16079					:10/INT, EXP>2**15, FID	
16080	103204	012737	000000	003030	MOV @0,@FLAG	;NO INTERRUPT
16081	103212	004767	000252		JSR R7,SCDSUB	;DO TEST
16082	103216	043300	000000	000000	.WORD 43300,0,0,0	;ACO
	103224	000000				
16083	103226	000000	014000		.WORD 0,14000	;RESULT
16084	103232	047700			.WORD 47700	; TEST FPS
16085	103234	047700			.WORD 47700	;RESULT FPS
16086					:11/INT, EXP>2**15, FIC=0	
16087	103236	012737	000000	003030	MOV @0,@FLAG	;NO INTERRUPT
16088	103244	004767	000220		JSR R7,SCDSUB	;DO TEST
16089	103250	143300	177777	177777	.WORD 143300,-1,-1,-1	;ACO
	103256	177777				
16090	103260	177777	163741		.WORD -1,163741	;RESULT
16091	103264	007300			.WORD 7300	; TEST FPS
16092	103266	007310			.WORD 7310	;RESULT FPS
16093					:12/LONG, EXP>2**32, FID	
16094	103270	012737	000002	003030	MOV @2,@FLAG	;INTERRUPT
16095	103276	004767	000166		JSR R7,SCDSUB	;DO TEST
16096	103302	050100	000000	000000	.WORD 50100,0,0,0	;ACO
	103310	000000				
16097	103312	000000	000000		.WORD 0,0	;RESULT
16098	103316	047700			.WORD 47700	; TEST FPS
16099	103320	147705			.WORD 147705	;RESULT FPS

FLOATING POINT TESTS

```

16100 ;13/LONG, EXP>2**32, FIC=0
16101 103322 012737 000000 003030 MOV    #0,#@FLAG ;NO INTERRUPT
16102 103330 004767 000134 JSR    R7,SCDSUB ;DO TEST
16103 103334 050377 177777 177777 .WORD 50377, 1, 1,-1 ;ACO
      103342 177777
16104 103344 000000 000000 .WORD 0,0 ;RESULT
16105 103350 007300 .WORD 7300 ; TEST FPS
16106 103352 007305 .WORD 7305 ;RESULT FPS
16107 ;14/LONG, EXP<0
16108 103354 005037 003030 CLR    @FLAG ;NO INTERRUPTS
16109 103360 004767 000104 JSR    R7,SCDSUB ;DO TEST
16110 103364 100200 177777 177777 .WORD 100200, 1, 1,-1 ;ACO
      103372 177777
16111 103374 000000 000000 .WORD 0,0 ;RESULT
16112 103400 007757 .WORD 7757 ; TEST FPS
16113 103402 007744 .WORD 7744 ;RESULT FPS
16114 ;15/INT, EXP<0
16115 103404 005037 003030 CLR    @FLAG ;NO INTERRUPTS
16116 103410 004767 000054 JSR    R7,SCDSUB ;DO TEST
16117 103414 037700 177777 177777 .WORD 37700,-1, 1,-2 ;ACO
      103422 177776
16118 103424 000000 177777 .WORD 0,-1 ;RESULT
16119 103430 007600 .WORD 7600 ; TEST FPS
16120 103432 007604 .WORD 7604 ;RESULT FPS
16121 ;16/INT, EXP=10
16122 103434 005037 003030 CLR    @FLAG ;NO INTERRUPTS
16123 103440 004767 000024 JSR    R7,SCDSUB ;DO TEST
16124 103444 004377 177777 177777 .WORD 4377,-1, 1,-1 ;ACO
      103452 177777
16125 103454 000000 177777 .WORD 0,-1 ;RESULT
16126 103460 007600 .WORD 7600 ; TEST FPS
16127 103462 007604 .WORD 7604 ;RESULT FPS
16128 ;
16129 103464 000167 000244 JMP    HOP21 ;GET OVER SUBROUTINE
16130 ;*****
16131 ;*****
16132 ;STCDI, STCDL, STCFI, STCFL
16133 ;
16134 ; ACO
16135 ; RESULT
16136 ; FPS BEFORE EXECUTION
16137 ; FPS AFTER EXECUTION
16138 ; (FEC)
16139 ;
16140 ;TRAP ON CONVERSION FAILURE
16141 ;*****
16142 ;*****
16143 ;
16144 103470 012602 SCDSUB: MOV    (SP)+,R2 ; RETURN ADDRESS TO USE AS POINTER
16145 103472 012737 103564 000244 MOV    #50@, @FPVEC ;REDIRECT TRAP VECTOR
16146 103500 012701 003144 MOV    @RECDST+2,R1 ;POINT TO RESULT AREA
16147 103504 012711 177777 MOV    #-1,(R1) ;PRELOAD RECEIVE DATA BUFFER
16148 103510 012741 177777 MOV    #-1,-(R1) ;
16149 103514 012700 000200 MOV    #200,R0 ;SET FPS TO DOUBLE
16150 103520 170100 LDFPS R0 ;
16151 103522 010204 MOV    R2,R4 ;POINT TO ACO DATA
16152 103524 172414 LDD    (R4),ACO ;LOAD ACO

```

FLOATING POINT TESTS

```

16153 103526 016200 000014      MOV    14(R2),R0      ;GET TEST FPS
16154 103532 170100             LDFPS  R0            ;LOAD TEST FPS
16155                          ;
16156 103534 175411      40$:  STCDI  ACO,(R1)  ;*TEST INSTRUCTION(ACCORDING TO MODE)
16157 103536 170327      1$:  STST   (PC)+    ;WAIT FOR POSSIBLE FPA TRAP.
16158 103540 000000             .WORD  0            ;STORE STATUS HERE.
16159                          ;
16160                          ;INSTRUCTION DIDNT TRAP
16161                          ;
16162 103542 032737 000001 003030  BIT    #1,#FLAG      ;VERIFY A NO TRAP CONDITION
16163 103550 001426             BEQ    2$            ;BRANCH IF GOOD
16164 103552 004737 140132      CALL   #DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16165 103556 104003             ERROR  +3           ;FPP ERROR
16166                          ;INSTRUCTION SHOULD HAVE TRAPPED
16167 103560 000167 000042      JMP    2$            ;REJOIN CODE
16168                          ;
16169                          ;INSTRUCTION TRAPPED
16170                          ;
16171 103564 032737 000001 003030 50$:  BIT    #1,#FLAG      ;SEE IF EXPECTING A TRAP
16172 103572 001005             BNE    51$          ;BRANCH IF EXPECTING A TRAP
16173 103574 004737 140132      CALL   #DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16174 103600 104003             ERROR  +3           ;FPP ERROR
16175                          ;INSTRUCTION WASNT SUPPOSE TO TRAP
16176 1C3602 000167 000020      JMP    2$            ;REJOIN CODE
16177 103606 012604      51$:  MOV    (SP)+,R4    ;SEE IF PC = INSTRUCTION
16178 103610 005726             TST   (SP)+        ;CLEAN UP STACK
16179 103612 022704 103536      CMP    #1$,R4      ;
16180 103616 001403             BEQ    2$            ;BRANCH IF GOOD COMPARE
16181 103620 004737 140132      CALL   #DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16182 103624 104003             ERROR  +3           ;FPP ERROR
16183                          ;PC WAS INCORRECT
16184                          ;
16185                          ;COMMON CODE FOR TRAP AND NO TRAP
16186                          ;VERIFY STATUS
16187                          ;
16188 103626 170203      2$:  STFPS  R3            ;SAVE FPS
16189 103630 016200 000016      MOV    16(R2),R0    ;GET EXPECTED STATUS
16190 103634 020003             CMP    R0,R3        ;VERIFY STATUS
16191 103636 001403             BEQ    3$            ;BRANCH IF GOOD
16192 103640 004737 140132      CALL   #DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16193 103644 104003             ERROR  +3           ;FPP ERROR
16194                          ;BAD FPS
16195 103646 010204      3$:  MOV    R2,R4        ;POINT TO EXPECTED DATA
16196 103650 062704 000010      ADD    #10,R4
16197 103654 004767 034202      4$:  JSR    R7,DATVFR  ;VERIFY DATA
16198 103660 005767 077234             TST   COUNT
16199 103664 001403             BEQ    5$            ;BRANCH IF GOOD
16200 103666 004737 140132      CALL   #DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16201 103672 104003             ERROR  +3           ;FPP ERROR
16202                          ;BAD ACO
16203 103674 005737 003030      5$:  TST    #FLAG      ;SEE IF NEED TO CHECK FEC
16204 103700 001002             BNE    7$            ;BRANCH IF NEED TO CHECK
16205 103702 000162 000020      JMP    20(R2)       ;RETURN FROM TEST
16206                          ;VERIFY FEC
16207 103706 012704 003122      7$:  MOV    #RECFEC,R4 ;POINT TO FEC AREA
16208 103712 170314             STST  (R4)          ;SAVE FEC
16209 103714 021427 000006      CMP    (R4),#6     ;VERIFY FEC FOR OVERFLOW

```

FLOATING POINT TESTS

```

16210 103720 001403          BEQ      8$          ;BRANCH IF GOOD
16211 103722 004737 140132  CALL    8@DETFPA    ;DETERMINE FLOATING POINT FAULT. $$$
16212 103726 104003          ERROR   +3          ;FPP ERROR
16213                                     ;BAD FEC
16214 103730 000162 000020  8$:    JMP      20(R2) ;RETURN FROM TEST
16215                                     ;
16216 103734          ;HOP21:
16217                                     ;
16220                                     ;
16221                                     ;
16222                                     ;-----
16223                                     ;TEST STCFI, STCFL
16224 103734          ;MSCF:
16225                                     ;
16226                                     ;1/LONG EXP =30
16227 103734 005037 003030  CLR     8@FLAG      ;NO INTERRUPTS
16228 103740 004767 177524  JSR     R7,SCDSUB   ;DO TEST
16229 103744 044541 052525 177777  .WORD  44541,52525,-1,1 ;ACO
16230 103754 000003 102525  .WORD  3,102525     ;RESULT
16231 103760 007517          .WORD  7517         ;TEST FPS
16232 103762 007500          .WORD  7500         ;RESULT FPS
16233                                     ;2/INT, EXP<0
16234 103764 005037 003030  CLR     8@FLAG      ;NO INTERRUPTS
16235 103770 004767 177474  JSR     R7,SCDSUB   ;DO TEST
16236 103774 002300 177777 177777  .WORD  2300,-1,-1,-1 ;ACO
16237 104004 000000 177777  .WORD  0,-1         ;RESULT
16238 104010 007400          .WORD  7400         ;TEST FPS
16239 104012 007404          .WORD  7404         ;RESULT FPS
16240                                     ;3/LONG, EXP
16241 104014 012737 000001 003030  MOV     81,8@FLAG   ;INTERRUPT
16242 104022 004767 177442  JSR     R7,SCDSUB   ;DO TEST
16243 104026 070000 177777 177777  .WORD  70000,-1,-1,-1 ;ACO
16244 104036 000000 000000  .WORD  0,0          ;RESULT
16245 104042 007540          .WORD  7540         ;TEST FPS
16246 104044 107545          .WORD  107545       ;RESULT FPS
16247                                     ;4/INT,EXP=5, FIC=0, FID=1
16248 104046 005037 003030  CLR     8@FLAG      ;NO INTERRUPTS
16249 104052 004767 177412  JSR     R7,SCDSUB   ;DO TEST
16250 104056 052000 000000 177777  .WORD  52000,0,-1,1  ;ACO
16251 104066 000000 177777  .WORD  0,-1         ;RESULT
16252 104072 047000          .WORD  47000        ;TEST FPS
16253 104074 047005          .WORD  47005        ;RESULT FPS
16254                                     ;
16257                                     ;
16258                                     ;-----
16259                                     ;TEST STEXP
16260                                     ;
16261 104076          ;MSXP:
16262                                     ;
16263                                     ;1/EXP=100
16264 104076 004767 000154          JSR     R7,SXPSUB   ;DO TEST
16265 104102 020000 000000 000000  .WORD  20000,0,0,0  ;ACO
104110 000000

```

FLOATING POINT TESTS

```

16266 104112 177700          .WORD -100          ;RESULT
16267 104114 007740          .WORD 7740          ; TEST FPS
16269 104116 007750          .WORD 7750          ;RESULT FPS
16269          ;2/EXP=201 FLOAT, NEG
16270 104120 004767 000132    JSR R7,SXPSUB          ;DO TEST
16271 104124 140377 177777 177777 .WORD 140377,-1,-1.0  ;ACO
16272 104134 000000          .WORD 1              ;RESULT
16273 104136 007500          .WORD 7500           ; TEST FPS
16274 104140 007500          .WORD 7500           ;RESULT FPS
16275          ;3/EXP=-177
16276 104142 004767 000110    JSR R7,SXPSUB          ;DO TEST
16277 104146 000177 177777 177777 .WORD 177,-1,-1,-1    ;ACO
16278 104154 177777          .WORD 177600         ;RESULT
16279 104160 007700          .WORD 7700           ; TEST FPS
16280 104162 007710          .WORD 7710           ;RESULT FPS
16281          ;4/EXP=-100
16282 104164 004767 000066    JSR R7,SXPSUB          ;DO TEST
16283 104170 020000 000000 177777 .WORD 20000,0,-1,-1  ;ACO
16284 104176 177777          .WORD -100           ;RESULT
16285 104200 177700          .WORD 40200          ; TEST FPS
16286 104202 040200          .WORD 40210          ;RESULT FPS
16287          ;5/EXP=200
16288 104206 004767 000044    JSR R7,SXPSUB          ;DO TEST
16289 104212 040000 000000 000000 .WORD 40000,0,0,0    ;ACO
16290 104220 000000          .WORD 0              ;RESULT
16291 104222 000000          .WORD 7700           ; TEST FPS
16292 104224 007700          .WORD 7704           ;RESULT FPS
16293          ;6/EXP=0
16294 104230 004767 000022    JSR R7,SXPSUB          ;DO TEST
16295 104234 000177 177777 177777 .WORD 177,-1,-1,-1    ;ACO
16296 104242 177777          .WORD 177600         ;RESULT
16297 104244 177600          .WORD 0              ; TEST FPS
16298 104246 000000          .WORD 10             ;RESULT FPS
16299          ;
16300 104252 000167 000120    JMP HOP22             ;GET OVER SUBROUTINE
16301          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16302          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16303          ;STEXP
16304          ;
16305          ; ACO
16306          ; EXPONENT RESULT
16307          ; FPS BEFORE EXECUTION
16308          ; FPS AFTER EXECUTION
16309          ;
16310          ;NO TRAPS CAN OCCUR
16311          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16312          ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16313 104256 012602          SXPSUB: MOV (SP)+,R2          ; RETURN ADDRESS TO USE AS POINTER
16314 104260 012737 104360 000244 MOV #50#,R0           ;REDIRECT TRAP VECTOR
16315 104266 012701 003142    MOV #RECDST,R1        ;POINT TO RESULT AREA
16316 104272 012700 000200    MOV #200,R0           ;SET FPS TO DOUBLE
16317 104276 170100          LDFPS R0              ;

```

FLOATING POINT TESTS

```

16318 104300 010204          MOV    R2,R4          ;POINT TO ACO DATA
16319 104302 172414          LDD   (R4),ACO       ;LOAD ACO
16320 104304 016200 000012  MOV   12(R2),R0     ;GET TEST FPS
16321 104310 170100          LDFPS R0             ;LOAD TEST FPS
16322
16323 104312 175011          40$:  STEXP  ACO,(R1) ;*TEST INSTRUCTION(ACCORDING TO MODE)
16324
16325          ;VERIFY STATUS
16326
16327 104314 170203          2$:  STFPS  R3        ;SAVE FPS
16328 104316 016200 000014  MOV   14(R2),R0     ;GET EXPECTED STATUS
16329 104322 020003          CMP   R0,R3        ;VERIFY STATUS
16330 104324 001403          BEQ   3$           ;BRANCH IF GOOD
16331 104326 004737 140132  CALL  @DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16332 104332 104003          ERROR +3          ;FPP ERROR
16333
16334 104334 016204 000010  3$:  MOV   10(R2),R4 ;POINT TO EXPECTED EXPONENT
16335 104340 020437 003142  CMP   R4,@RECDST   ;VERIFY EXPONENT
16336 104344 001403          BEQ   5$           ;BRANCH IF GOOD
16337 104346 004737 140132  CALL  @DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16338 104352 104003          ERROR +3          ;FPP ERROR
16339
16340 104354 000162 000016  5$:  JMP   16(R2)    ;RETURN FROM TEST
16341
16342          ;INSTRUCTION TRAPPED
16343
16344 104360 012600          50$: MOV   (SP)+,R0   ;SAVE PC
16345 104362 012605          MOV   (SP)+,R5     ;SAVE OLD PS
16346 104364 004737 140132  CALL  @DEFPA       ;DETERMINE FLOATING POINT FAULT. $$$
16347 104370 104003          ERROR +3          ;FPP ERROR
16348
16349 104372 000167 177756  JMP   5$           ;WILD TRAP DURING STEXP
16350
16351 104376          ;REJOIN CODE
16352
16353          ;
16354          ;
16355          ;
16356          ;
16357          ;
16358          ;
16359          ;
16360          ;
16361          ;
16362          ;
16363          ;
16364          ;
16365          ;
16366          ;
16367          ;
16368          ;
16369          ;
16370          ;
16371          ;
16372          ;
16373          ;
16374          ;
16375          ;
16376          ;

```

TEST - CHECK REGISTER ACCESS

```

16377 ;EXIT TST
16378 ;
16379 ;ADDRESS TABLE: 17777746
16380 ; 17777744
16381 ; 17777752
16382 ;
16383 ;ENDTST
16384 ;ERROR ROUTINE: SET TIMEOUT FLAG
16385 ; RETURN
16386 ;
16387 ;*****
104376 000004
16388 104400 000240
16389 104402 005737 003032
16390 104406 001002
16391
16392 104410 000240
16393 104412 000453
16394 104414 000240
16395 104416 012737 137542 000004
16396 104424 013737 000004 003012
16397 104432 012737 104536 000004
16398 104440 005001
16399 104442 005737 177746
16400 104446 005701
16401 104450 001404
16402 104452 012737 177746 001122
16403 104460 104130
16404 104462 005001
16405 104464 005737 177744
16406 104470 005701
16407 104472 001404
16408 104474 012737 177744 001122
16409 104502 104130
16410 104504 005001
16411 104506 005737 177752
16412 104512 005701
16413 104514 001404
16414 104516 012737 177752 001122
16415 104524 104130
16416 104526 013737 003012 000004
16417 104534 000402
16418
16419 104536
16420 104536 005201
16421 104540 000002
16422

```

```

TST2: SCOPE
NOP
TST CCHPAS ;have done enough inclusive passes?
BNE 99$ ; not yet

99$: NOP ; debug aid
BR TST3 ;;GO TO NEXT TEST

MOV #TOUT, #4
MOV #4, SLOC00 ;SAVE CONTENTS OF VECTOR 4
MOV #ERROUT, #4 ;LET VECTOR 4 POINT TO ERROR ROUTINE
CLR R1 ;CLEAR TIMEOUT FLAG
TST CCR ;READ REGISTER UNDER TEST
TST R1 ;TIMEOUT?
BEQ 2$
MOV #CCR, #BDADR ;STORE LOCATION
ERROR +130 ;TIMEOUT ACCESSING CCR
2$: CLR R1 ;CLEAR TIMEOUT FLAG
TST MSER ;READ MSER
TST R1 ;TIMEOUT ?
BEQ 3$
MOV #MSER, #BDADR ;STORE LOCATION
ERROR +130 ;TIMEOUT ACCESSING MSER
3$: CLR R1 ;CLEAR TIMEOUT FLAG
TST HITMIS ;ACCESS HIT/MISS
TST R1 ;TIMEOUT?
BEQ 4$
MOV #HITMIS, #BDADR ;STORE LOCATION
ERROR +130 ;TIMEOUT ACCESSING HIT/MISS
4$: MOV SLOC00, #4 ;RESTORE VECTOR 4
BR TST3 ;;GO TO NEXT TEST

ERROUT:
INC R1 ;FLAG TIMEOUT
RTI

```


TEST - CCR REGISTER BIT TEST

```

16424 .SBTTL TEST - CCR REGISTER BIT TEST
16425 ;CCR REGISTER BIT TEST - THIS TEST WILL VERIFY THAT EACH READ/WRITE BIT OF
16426 ;THE CACHE CONTROL REGISTER CAN BE SET AND CLEARED INDIVIDUALLY AND THAT
16427 ;BITS 15 - 11 AND BIT 8 ARE ALWAYS READ AS ZEROS.
16428 ;
16429 ;BGNTST
16430 ;INITIALIZE GOOD DATA TO #1
16431 ;CLEAR CCR
16432 ;DO UNTIL ALL BITS TESTED
16433 ;. WRITE GOOD DATA TO CCR
16434 ;. READ CCR
16435 ;. IF CCR NOT EQUAL TO GOOD DATA THEN
16436 ;. . IF CCR EQUAL TO ZERO THEN
16437 ;. . . IF GOOD DATA NOT EQUAL TO BIT 15-11 OR BIT 8 THEN
16438 ;. . . . ERROR IN READ/WRITE BITS OF CCR
16439 ;. . . . ENDIF
16440 ;. ENDF
16441 ;. UPDATE TO NEXT BIT
16442 ;ENDDO
16443 ;ENDTST
16444
16445 ;:*****
TST3: SCOPE
      NOP
16446 104544 000240      TST      CCHPAS          ;have done enough inclusive passes?
16447 104546 005737 003032 BNE      99$          ; not yet
16448 104552 001002      NOP
16449 104554 000240      NOP          ; debug aid
16450 104556 000431      BR       TST4          ;;GO TO NEXT TEST
16451 104560 000240 99$:  NOP
16452 104562 012701 000001 MOV      #1,      R1          ;INITIALIZE GOOD DATA TO #1
16453 104566 005037 177746 CLR      CCR          ;CLEAR CCR
16454 104572 042737 001000 177520 BIC      #1000,BCSR ;DISABLE HALT ON BREAK
16455 104600 010137 177746 1$:  MOV      R1,      CCR          ;WRITE GOOD DATA TO CCR
16456 104604 023701 177746 CMP      CCR,     R1          ;IF CCR NOT EQUAL GOOD DATA
16457 104610 001407      BEQ      3$          ;THEN
16458 104612 005737 177746 TST      CCR          ;IF CCR EQUAL TO ZERO
16459 104616 001003      BNE      2$          ;THEN
16460 104620 032701 174400 BIT      #174400,R1 ;IF GOOD DATA NE TO BIT 15-11 OR BIT 8
16461 104624 001001      BNE      3$          ;THEN
16462 104626 104004 2$:  ERROR  +4          ;ERROR IN READ/WRITE BITS OF CCR
16463 104630 006101 3$:  ROL      R1          ;UPDATE TO NEXT BIT
16464 104632 103362      BCC      1$          ;DO UNTIL ALL BITS TESTED
16465 104634 052737 001000 177520 BIS      #1000,BCSR ;ENABLE HALT ON BREAK
16466
16467

```


TEST - FORCE MISS TEST

```

16526      ;.      .      IF HIT THEN
16527      ;.      .      ERROR FORCE MISS READS FROM CACHE
16528      ;.      .      ELSE
16529      ;.      .      ERROR FORCE MISS READS FROM CACHE AND MISS
16530      ;.      .      ENDIF
16531      ;.      ELSE
16532      ;.      .      ERROR IN DATA PATH
16533      ;.      ENDIF
16534      ;ENDIF
16535      ;CLEAR CCR<3:2>
16536      ;WRITE TEST ADDRESS
16537      ;ENDTST
16538      ;
16539      ;*****
TST4:      SCOPE
           NOP
16540      104642 000004      TST      CCHPAS      ;have done enough inclusive passes?
16541      104644 000240      BNE      99$      ; not yet
16542      104646 005737 003032      NOP      ; debug aid
16543      104652 001002      BR       TST5      ;;GO TO NEXT TEST
16544      104654 000240      99$:    NOP
16545      104656 000563      MOV      @#114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
16546      104660 000240      MOV      #FMPARR,@#114 ;SETUP PARITY VECTOR TO POINT TO HANDLER
16547      104662 013737 000114 003012      CLR      R3      ;CLEAR MSER SAVE LOCATION
16548      104670 012737 105214 000114      CLR      @#TSTLOC ;WRITE TEST LOCATION WITH PATTERN 1
16549      104700 005037 003162      MOV      @BIT02, CCR ;SET CCR BITS<3:2> = 0,1
16550      104704 012737 000004 177746      MOV      #177777,@#TSTLOC ;WRITE TEST LOCATION WITH PATTERN 2
16551      104712 012737 177777 003162      MOV      #200, CCR ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
16552      104720 012737 000200 177746      MOV      #177777,$GDDAT ;SAVE DATA IN MEMORY
16553      104726 012737 001124      BIC      #1000,BCSR ;DISABLE HALT ON BREAK
16554      104734 042737 001000 177520      MOV      @#TSTLOC,R1 ;READ TEST ADDRESS
16555      104742 013701 003162      MOV      @#HITMIS,R2 ;SAVE HIT/MISS DATA
16556      104746 013702 177752      BIS      #1000,BCSR ;ENABLE HALT ON BREAK
16557      104752 052737 001000 177520      TST      R1 ;COMPARE RECEIVED DATA TO PATTERN 1
16558      104760 005701      BEQ      1$ ;IF DATAS NOT EQUAL THEN
16559      104762 001451      CMP      #177777,R1 ;IF DATA EQUAL TO PATTERN 2
16560      104764 022701 177777      BNE      106$ ;THEN
16561      104770 001045      BIT      @#BIT01, R2 ;IF HIT
16562      104772 032702 000002      BEQ      100$ ;THEN
16563      104776 001402      ERROR   +5 ;FORCE MISS WRITES TO CACHE
16564      105000 104005      BR       1$ ;ELSE
16565      105002 000441      BIT      #100340,R3 ;IF PARITY INDICATORS EQUAL 0
16566      105004 032703 100340      100$:   BNE      101$ ;THEN
16567      105010 001002      ERROR   +6 ;FORCE MISS WRITE INVALIDATES CACHE
16568      105012 104006      BR       1$ ;ELSE
16569      105014 000434      CMP      #100340,R3 ;IF ALL PARITY INDICATORS SET
16570      105016 022703 100340      101$:   BNE      102$ ;THEN
16571      105022 001002      ERROR   +7 ;PARITY ABORT AFTER FORCE MISS
16572      105024 104007      BR       1$ ;ELSE
16573      105026 000427      BIT      @#BIT05, R3 ;IF TAG PARITY ERROR
16574      105030 032703 000040      102$:   BEQ      103$ ;THEN
16575      105034 001402      ERROR   +10 ;TAG PARITY ERROR AFTER FORCE MISS
16576      105036 104010      BR       1$ ;ELSE
16577      105040 000422      MOV      R3, R4 ;COPY MSER INTO REG 4
16578      105042 010304      103$:   BIC      #177477,R4 ;MASK FOR B0 AND B1 DATA
16579      105044 042704 177477      CMP      #300, R4 ;IF B0 AND B1 DATA
16580      105050 022704 000300      BNE      104$ ;THEN
16581      105054 001002

```

TEST - FORCE MISS TEST

```

16582 105056 104011          ERROR +11          ;DATA PARITY ERROR AFTER FORCE MISS
16583 105060 000412          BR 1$           ;ELSE
16584 105062 032703 000100 104$: BIT #100, R3    ;IF B0 ERROR
16585 105066 001401          BEQ 105$        ;THEN
16586 105070 104012          ERROR +12        ;LOW BYTE PARITY ERROR AFTER FORCE MISS
16587 105072 032703 000200 105$: BIT #200, R3    ;IF B1 ERROR
16588 105076 001403          BEQ 1$           ;THEN
16589 105100 104013          ERROR +13        ;HIGH BYTE PARITY ERROR AFTER FORCE MISS
16590 105102 000401          BR 1$           ;ENDIF
16591 105104 104014          106$: ERROR +14    ;ERROR DATA PATH
16592 105106 005003          1$: CLR R3         ;CLEAR MSER SAVE REGISTER
16593 105110 052737 000010 177746 BIS #BIT03, CCR  ;SET CCR BITS <3:2> = 1,0
16594 105116 042737 001000 177520 BIC #1000,BCSR  ;DISABLE HALT ON BREAK
16595 105124 013701 003162      MOV #TSTLOC,R1  ;READ TEST LOCATION
16596 105130 013702 177752      MOV #HITMIS,R2  ;SAVE HIT/MISS REGISTER DATA
16597 105134 052737 001000 177520 BIS #1000,BCSR  ;ENABLE HALT ON BREAK
16598 105142 020127 177777      CMP R1, #177777 ;COMPARE RECEIVED DATA TO PATTERN 2
16599 105146 001412          BEQ 2$           ;IF DATAS NOT EQUAL THEN
16600 105150 005701          TST R1           ;IF RECIEVED DATA EQUAL TO PATTERN 1
16601 105152 001007          BNE 108$        ;THEN
16602 105154 032702 000002      BIT #BIT01, R2  ;IF HIT
16603 105160 001402          BEQ 107$        ;THEN
16604 105162 104015          ERROR +15        ;FORCE MISS READS FROM CACHE
16605 105164 000403          BR 2$           ;ELSE
16606 105166 104016          107$: ERROR +16    ;FORCE MISS READS FROM CACHE AND MISS
16607 105170 000401          BR 2$           ;ENDIF
16608 105172 104014          108$: ERROR +14    ;ERROR IN DATA PATH
16609 105174 013737 003012 000114 2$: MOV SLOC00, #114 ;RESTORE VECTOR 114
16610 105202 005037 177746      CLR CCR         ;CLEAR CCR BITS<3:2>
16611 105206 005037 003162      CLR #TSTLOC    ;WRITE TEST ADDRESS
16612 105212 000405          BR TST5         ;;GO TO NEXT TEST
16613
16614
16615 105214 011637 001122      FMPARR: MOV (SP),#BDADR ;SAVE ADDRESS THAT CAUSED ABORT
16616 105220 013703 177744      MOV MSER, R3   ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
16617 105224 000002          RTI                    ;REGISTER AND RETURN
16618
16619
16620          .SBTTL TEST - HIT/MISS REGISTER TEST PART 1
16621          ;HIT/MISS REGISTER TEST PART 1 - THIS TEST WILL VERIFY THAT THE HIT/MISS
16622          ;REGISTER CORRECTLY LOGS HITS AND MISSES.FIRST WRITE A TEST ADDRESS WITH
16623          ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN INTO
16624          ;THE CACHE. THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH NEW DATA.
16625          ;NEXT CLEAR CCR BITS<3:2> AND READ THE TEST ADDRESS. THE HIT/MISS REGISTER
16626          ;SHOULD LOGGED A HIT. FINALLY READ THE TEST ADDRESS PLUS 20000(8) THE
16627          ;HIT/MISS REGISTER SHOULD HAVE LOGGED A MISS.
16628          ;
16629          ;
16630          ;BGNTST
16631          ;WRITE TEST ADDRESS WITH PATTERN 1
16632          ;SET CCR BITS<3:2> = 0,1
16633          ;WRITE TEST ADDRESS WITH PATTERN 2
16634          ;CLEAR CCR BIT<3>
16635          ;READ TEST ADDRESS
16636          ;IF HIT/MISS REGISTER BIT 1 NOT SET THEN
16637          ;. ERROR IN RECORDING HITS IN HIT/MISS
16638          ;ENDIF

```

TEST - HIT/MISS REGISTER TEST PART 1

```

16639 ;READ TEST ADDRESS + 20000(8)
16640 ;IF HIT/MISS REGISTER BIT 1 SET THEN
16641 ;. ERROR IN RECORDING HITS IN HIT/MISS
16642 ;ENDIF
16643 ;ENDTST
16644 ;
16645 ;*****
105226 000004 TST5: SCOPE
16646
16647 105230 000240 NOP
16648 105232 005737 003032 TST CCHPAS ;have done enough inclusive passes?
16649 105236 001002 BNE 99$ ; not yet
16650 105240 000240 NOP ; debug aid
16651 105242 000463 BR TST6 ;;GO TO NEXT TEST
16652 105244 000240 99$: NOP
16653 105246 013737 000114 003012 MOV @#114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
16654 105254 012737 105404 000114 MOV @HMPARR,@#114 ;SETUP PARITY VECTOR TO POINT TO HANDLER
16655 105262 005037 003162 CLR @TSTLOC ;WRITE TEST LOCATION WITH PATTERN 1
16656 105266 012737 000004 177746 MOV @BIT02, CCR ;SET CCR BITS<3:2> = 0,1
16657 105274 012737 177777 003162 MOV @177777,@TSTLOC ;WRITE TEST LOCATION WITH PATTERN 2
16658 105302 012737 000200 177746 MOV @200, CCR ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
16659 105310 042737 001000 177520 BIC @1000,BCSR ;DISABLE HALT ON BREAK
16660 105316 013701 003162 MOV @TSTLOC,R1 ;READ TEST ADDRESS
16661 105322 013737 177752 114150 MOV HITMIS,RECDAT ;STORE REGISTER
16662 105330 032737 000004 114150 BIT @BIT02,RECDAT ;IF HIT/MISS REGISTER BIT 1 NOT SET
16663 105336 001001 BNE 1$ ;THEN
16664 105340 104045 ERROR +45 ;ERROR IN RECORDING HITS IN HIT/MISS
16665 105342 013701 023162 1$: MOV @TSTLOC+8192.,R1 ;READ TEST LOCATION + 20000(8)
16666 105346 013737 177752 114150 MOV HITMIS,RECDAT ;STORE REGISTER
16667 105354 032737 000004 114150 BIT @BIT02,RECDAT ;IF HIT/MISS REGISTER BIT 1 SET
16668 105362 001401 BEQ 2$ ;THEN
16669 105364 104045 ERROR +45 ;ERROR IN RECORDING HITS IN HIT/MISS
16670 105366 013737 003012 000114 2$: MOV SLOC00, @#114 ;RESTORE VECTOR 114
16671 105374 052737 001000 177520 BIS @1000,BCSR ;ENABLE HALT ON BREAK
16672 105402 000403 BR TST6 ;;GO TO NEXT TEST
16673
16674 105404 013703 177744 HMPARR: MOV MSER, R3 ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
16675 105410 000002 RTI ;REGISTER AND RETURN
16676

```

TEST - HIT/MISS REGISTER TEST

```

16678 .SBTTL TEST - HIT/MISS REGISTER TEST
16679 ;HIT/MISS REGISTER TEST - THIS TEST WILL VERIFY THAT THE HIT/MISS
16680 ;REGISTER CORRECTLY LOGS CACHE HITS AND MISSES. IT WILL ALSO VERIFY
16681 ;THAT EACH BIT OF THE REGISTER IS UNIQUE. THIS WILL BE DONE BY
16682 ;FLOATING A ZERO THROUGH A FIELD OF ONES. THE ROTATING WILL BE DONE
16683 ;BY EXECUTING A TST @HM (WHERE HM IS THE ADDRESS OF THE HIT/MISS
16684 ;REGISTER) AT SUCCESSIVE POSITIONS IN A SET OF EIGHT NOPS. THE READ
16685 ;OF THE I/O PAGE ADDRESS OF THE HIT/MISS REGISTER SHOULD CAUSE A
16686 ;MISS TO BE RECORDED.
16687 ;
16688 ;BGNTST
16689 ;INITIALIZE LOOP INDICATOR
16690 ;INITIALIZE EXPECTED DATA
16691 ;DO UNTIL LOOP INDICATOR = SEVEN
16692 ;. PUT BACKGROUND DATA INTO EXECUTION BUFFER
16693 ;. PUT TST INSTRUCTION INTO TEST LOCATION
16694 ;. JUMP TO EXECUTE BUFFER
16695 ;. IF EXPECTED DATA NE RECEIVED DATA THEN
16696 ;. ERROR IN RECORDING HITS IN HIT/MISS
16697 ;. ENDF
16698 ;. INCREMENT LOOP INDICATOR
16699 ;. UPDATE EXPECTED DATA
16700 ;ENDDO
16701 ;EXIT TST
16702 ;
16703 ;BACKGROUND DATA:NOP
16704 ; NOP
16705 ; NOP
16706 ; NOP
16707 ; NOP
16708 ; NOP
16709 ; NOP
16710 ; NOP
16711 ; MOV @HM,R2
16712 ; RTS PC
16713 ;ENDTST
16714 ;*****
16715 105412 000004 TST6: SCOPE
16716 105414 000240 NOP
16717 105416 005737 003032 TST CCHPAS ;have done enough inclusive passes?
16718 105422 001002 BNE 99$ ; not yet
16719 105424 000240 NOP ; debug aid
16720 105430 000240 BR TST7 ;GO TO NEXT TEST
16721 105432 005037 003154 99$: NOP
16722 105436 013737 023204 001162 CLR LOOPIN ;INITIALIZE LOOP INDICATOR
16723 105444 012703 003164 MOV @TSTLOC+20022,@TMP1 ;SAVE FOR LATER
16724 105450 012704 105634 MOV @TSTLOC+2,R3 ;GET ADDRESS OF EXECUTION BUFFER
16725 105454 012705 177752 MOV @EXPTBL,R4 ;GET ADDRESS OF EXPECTED DATA TABLE
16726 105460 042737 001000 177520 MOV @HITMIS,R5 ;PUT ADDRESS OF HIT/MISS REGISTER IN GPR
16727 105466 023727 003154 000007 1$: BIC @1000,BCSR ;DISABLE HALT ON BREAK
16728 105474 001440 BEQ ENDRHT ;DO UNTIL LOOP INDICATOR EQUALS 7
16729 105476 012702 000013 MOV @13,R2 ;THEN EXIT
16730 105502 012700 105606 MOV @BACDAT,R0 ;PUT BACKGROUND DATA INTO
16731 105506 012701 003162 MOV @TSTLOC,R1 ;EXECUTION BUFFER
16732 105512 012021 2$: MOV (R0), (R1)
16733 105514 077202 SOB R2, 2$ ;LOOP FOR TEN WORDS

```

TEST - HIT/MISS REGISTER TEST

```

16734 105516 023727 003154 000006      CMP      LOOPIN, #6          ;IS THIS LAST LOOP
16735 105524 001004                    BNE      3$                 ;IF YES THEN
16736 105526 013737 001162 023204      MOV      $TMP1, @TSTLOC+2002 ;INVALIDATE FETCH OF "RECDAT"
16737 105534 000404                    BR       4$                 ;ELSE
16738 105536 012723 005737      3$:    MOV      #5737, (R-)+    ;MOVE "TST" INSTRUCTION TO BUFFER
16739 105542 012713 177752      MOV      @HITMIS, (R5)     ;MOVE HIT.MISS REG. ADD. TO BUFFER
16740 105546 004737 003162      4$:    JSR      PC, TSTLOC   ;GO EXECUTE TEST CODE
16741 105552 021437 114150      CMP      (R4), RECDAT     ;IF EXPECTED DATA NOT EQUAL TO
16742 105556 001403                    BEQ      5$                 ;RECEIVED DATA THEN
16743 105560 011437 001124      MOV      (R4), $GDDAT     ;SAVE EXPECTED PATTERN
16744 105564 104017                    ERROR   +17                ;ERROR IN RECORDING HITS IN HIT/MISS
16745 105566 005724      5$:    TST      (R4)+        ;INCREMENT POINTER TO EXPECTED PATTERN
16746 105570 005237 003154      INC      LOOPIN          ;INCREMENT LOOP COUNTER
16747 105574 000734                    BR       1$
16748 105576                    ENDHRT:
16749 105576 052737 001000 177520      BIS      #1000,BCSR       ;ENABLE HALT ON BREAK
16750 105604 000422                    BR       TST7              ;;GO TO NEXT TEST
16751
16752 105606 000240      BACDAT: .WORD  NOP
16753 105610 000240      .WORD  NOP
16754 105612 000240      .WORD  NOP
16755 105614 000240      .WORD  NOP
16756 105616 000240      .WORD  NOP
16757 105620 000240      .WORD  NOP
16758 105622 000240      .WORD  NOP
16759 105624 000240      .WORD  NOP
16760 105626 011537 114150      MOV      (R5),RECDAT     ;SAVE CONTENTS OF HIT/MISS REGISTER
16761 105632 000207      RTS      PC
16762
16763 105634 000077      EXPTBL: .WORD  77
16764 105636 000037      .WORD  37
16765 105640 000057      .WORD  57
16766 105642 000067      .WORD  67
16767 105644 000073      .WORD  73
16768 105646 000075      .WORD  75
16769 105650 000076      .WORD  76
16770

```

TEST - BYTE ALLOCATION TEST

```

16772 .SBTTL TEST - BYTE ALLOCATION TEST
16773 ;BYTE ALLOCATION TEST - THIS TEST WILL VERIFY THAT THE CACHE SYSTEM CORRECTLY
16774 ;HANDLES BYTE ACCESSES. THE TEST WILL FIRST CHECK THAT A WRITE ACCESS WHICH
16775 ;IS A MISS DOES NOT UPDATE THE CACHE. THIS WILL BE DONE BY FIRST ASSURING
16776 ;A MISS AT A TEST LOCATION. THEN A WRITE BYTE ACCESS WILL BE DONE. FINALLY
16777 ;THE LOCATION WILL BE READ. THE WRITE BYTE SHOULD NOT HAVE ALLOCATED THE
16778 ;ADDRESS. NEXT THE TEST WILL VERIFY THAT IF A WRITE BYTE ACCESS IS A HIT
16779 ;THAT THE CACHE LOCATION IS UPDATED. THE TEST WILL FIRST WRITE A TEST LOCATION
16780 ;WITH A PATTERN TO ALLOCATE THE ADDRESS. THEN A SECOND PATTERN WILL BE WRITTEN
16781 ;TO THE HIGH BYTE OF THE TEST LOCATION. THE TEST LOCATION WILL THEN BE READ.
16782 ;IF THE HIGH BYTE OF THE TEST LOCATION IS EQUAL TO THE SECOND PATTERN THEN THE
16783 ;LOCATION WAS UPDATED. THE SECOND TEST WILL BE REPEATED TO TEST LOW BYTE
16784 ;ACCESSES.
16785 ;
16786 ;BGNTST
16787 ;SAVE VECTOR 114
16788 ;LET VECTOR 114 POINT TO BYTE PARITY ROUTINE
16789 ;SET PARITY ABORT BIT IN CACHE CONTROL REGISTER
16790 ;READ TEST LOCATION + 4K
16791 ;WRITE LOW BYTE TO TEST ADDRESS
16792 ;READ LOW BYTE OF TEST ADDRESS
16793 ;IF HIT/MISS REGISTER BIT 1 SET THEN
16794 ;. ERROR WRITE BYTE ALLOCATES THE CACHE
16795 ;ENDIF
16796 ;WRITE PATTERN 1 TO TEST LOCATION
16797 ;WRITE BYTE PATTERN 2 TO TEST LOCATION-1
16798 ;IF BIT1 NOTSET IN HIT/MISS REGISTER THEN
16799 ;. IF B0 PARITY ERROR THEN
16800 ;. . ERROR LOW BYTE PARITY ERROR ON WRITE BYTE HIT
16801 ;. ELSE
16802 ;. . IF B1 PARITY ERROR THEN
16803 ;. . . ERROR HIGH BYTE PARITY ERROR ON WRITE BYTE HIT
16804 ;. . ELSE
16805 ;. . . WRITE BYTE HIT DOES NOT RECORD A HIT
16806 ;. . . ENDIF
16807 ;. . ENDIF
16808 ;ELSE
16809 ;. READ TEST LOCATION
16810 ;. IF RECEIVED DATA NOT EQUAL TO EXPECTED DATA THEN
16811 ;. . IF BYTE DATA REVERSED THEN
16812 ;. . . ERROR BYTES REVERSED ON WRITE CYCLES
16813 ;. . ELSE
16814 ;. . . IF HIGH BYTE DATA ZERO THEN
16815 ;. . . . ERROR IN WRITING TO HIGH BYTE
16816 ;. . . ELSE
16817 ;. . . . ERROR IN WRITING TO HIGH BYTE
16818 ;. . . . ENDIF
16819 ;. . . ENDIF
16820 ;. . ENDIF
16821 ;ENDIF
16822 ;WRITE PATTERN 1 TO TEST LOCATION
16823 ;WRITE BYTE PATTERN 2 TO TEST LOCATION LOW BYTE
16824 ;IF BIT1 NOTSET IN HIT/MISS REGISTER THEN
16825 ;. IF B0 PARITY ERROR THEN
16826 ;. . ERROR LOW BYTE PARITY
16827 ;. ELSE
16828 ;. . IF B1 PARITY ERROR THEN

```


TEST BYTE ALLOCATION TEST

```
16829          ::          .          .          ERROR HIGH BYTE PRITY
16830          ::          .          .          ELSE
16831          ::          .          .          WRITE BYTE HIT DOES NOT RECORD A HIT
16832          ::          .          .          ENDF
16833          ::          .          .          ENDF
16834          ::          .          .          ;ELSE
16835          ::          .          .          READ TEST LOCATION
16836          ::          .          .          IF RECIEVED DATA NOT EQUAL TO EXPECTED DATA THEN
16837          ::          .          .          IF BYTE DATA REVERSED THEN
16838          ::          .          .          ERROR BYTES REVERSED
16839          ::          .          .          ELSE
16840          ::          .          .          IF LOW BYTE DATA ZERO THEN
16841          ::          .          .          ERROR IN DATA PATH
16842          ::          .          .          ELSE
16843          ::          .          .          ERROR IN DATA PATH
16844          ::          .          .          ENDF
16845          ::          .          .          ENDF
16846          ::          .          .          ENDF
16847          ::          .          .          ;ENDIF
16848          ::          .          .          ;CLEAR CACHE CONTROL REGISTER
16849          ::          .          .          ;RESTORE VECTOR 114
16850          ::          .          .          ;EXIT TST
16851          ::          .          .          ;
16852          ::          .          .          ;BYTE PARITY ROUTINE:  SAVE MSER
16853          ::          .          .          ;                               RETURN
16854          ::          .          .          ;ENDTST
16855          ::          .          .          ;
16856          ::          .          .          ;*****
```

```
16857 105652 000004
16858 105654 000240
16859 105656 005737 003032
16860 105662 001002
16861 105664 000240
16862 105666 000517
16863 105670 000240
16864 105672 013737 000114 003012
16865 105700 012737 106114 000114
16866 105706 005003
16867 105710 012737 000200 177746
16868 105716 005737 023162
16869 105722 042737 001000 177520
16870 105730 105037 003162
16871 105734 105737 003162
16872 105740 013737 177752 114150
16873 105746 032737 000004 114150
16874 105754 001401
16875 105756 104020
16876 105760 005037 003162 1$:
16877 105764 152737 000377 003163
16878 105772 013737 177752 114150
16879 106000 032737 000004 114150
16880 106006 001002
16881 106010 104021 101$:
16882 106012 000405
16883 106014 022737 177400 003162 2$:
16884 106022 001401
16884 106024 104022

TST7:  SCOP=
      NOP
      TST CCHPAS ;have done enough inclusive passes?
      BNE 99$ ; not yet
      NOP ; debug aid
      BR TST10 ;;GO TO NEXT TEST
99$:  NOP
      MOV #114, SLOC00 ;SAVE VECTOR 114
      MOV #BYARR,#114 ;LET VECTOR 114 POINT TO ABORT ROUTINE
      CLR R3 ;CLEAR MSER SAVE REGISTER
      MOV #BIT07,CCR ;SET PARITY ABORT BIT IN CCR
      TST TSTLOC+8192. ;READ TEST LOCATION + 4K TO CAUSE MISS
      BIC #1000,BCSR ;DISABLE HALT ON BREAK
      CLRB TSTLOC ;WRITE LOW BYTE OF TEST LOCATION
      TSTB TSTLOC ;READ LOW BYTE OF TEST LOCATION
      MOV HITMIS,RECDAT ;STORE REGISTER
      BIT #BIT02,RECDAT ;IF BIT1 OF HIT/MISS REGISTER SET
      BEQ 1$ ;THEN
      ERROR +20 ;WRITE BYTE ALLOCATES CACHE
      CLR TSTLOC ;WRITE PATTERN 1 TO TEST LOCATION
      BISB #377, #TSTLOC+1 ;WRITE PATTERN 2 TO HIGH BYTE
      MOV HITMIS,RECDAT ;STORE REGISTER
      BIT #BIT02,RECDAT ;IF BIT1 NOTSET IN HIT/MISS REGISTER
      BNE 2$ ;THEN
      ERROR +21 ;WRITE BYTE HIT DOES NOT RECORD HIT
      BR 3$ ;ELSE
      CMP #177400,#TSTLOC ;IF TEST LOCATION NOT EQUAL TO
      BEQ 3$ ;PATTERN 2 THEN
      ERROR +22 ;BYTES REVERSED ON WRITE CYCLES
```

TEST - BYTE ALLOCATION TEST

```

16885 106026 005037 003162      3$: CLR      TSTLOC      ;WRITE PATTERN 1 TO TEST LOCATION
16886 106032 152737 000377 003162  BISB     #377,  @#TSTLOC ;WRITE PATTERN 2 TO LOW BYTE
16887 106040 013737 177752 114150  MOV     HITMIS,RECDAT ;STORE REGISTER
16888 106046 032737 000004 114150  BIT     #BIT02,RECDAT ;IF BIT1 NOTSETIN HIT/MISS REGISTER
16889 106054 001001                BNE     4$           ;THEN
16890 106056 104012                ERROR  +12          ;LOW BYTE PARITY ERROR ON WRITE BYTE HIT
16891 106060 022737 000377 003162  4$: CMP     #377,  @#TSTLOC ;IF TEST LOCATION NOT EQUAL TO
16892 106066 001401                BEQ     5$           ;PATTERN 2 THEN
16893 106070 104022                ERROR  +22          ;BYTES REVERSED ON WRITE CYCLES
16894 106072 005037 177746      5$: CLR     CCR         ;CLEAR CCR BEFORE EXIT
16895 106076 052737 001000 177520  BIS     #1000,BCSR    ;ENABLE HALT ON BREAK
16896 106104 013737 003012 000114  MOV     SLOC00, @#114 ;RESTORE VECTOR 114
16897 106112 000405                BR      TST10      ;;GO TO NEXT TEST
16898
16899
16900 106114 011637 001122      BYPARR: MOV    (SP),  $BDADR ;SAVE ADDRESS THAT CAUSE ABORT
16901 106120 013703 177744      MOV     MSER,   R3     ;SAVE CONTENTS OF MSER
16902 106124 000002                RTI                    ;RETURN
16903

```

TEST - PDR BIT15 (BYPASS) TEST

```

16905 .SBTTL TEST - PDR BIT15 (BYPASS) TEST
16906 ;PDR BIT15 (BYPASS) TEST - THIS TEST WILL VERIFY THAT WHEN BIT<15> IS SET
16907 ;IN A PDR AND AN ACCESS IS MADE TO A LOCATION MAPPED BY THE SELECTED PDR
16908 ;THAT WOULD NORMALLY CAUSE A CACHE ACCESS, THE CACHE IS BYPASSED (I.E. THE
16909 ;CACHE LOCATION IS INVALIDATED AND A MAIN MEMORY IS READ. THIS WILL BE DONE
16910 ;BY FIRST WRITING A TEST LOCATION WITH A KNOWN PATTERN TO ALLOCATE THE ADDRESS.
16911 ;THEN THE LOCATION WILL BE WRITTEN AGAIN WITH THE MMU ENABLED AND BIT <15> SET
16912 ;IN THE CONTROLLING PDR WITH A NEW PATTERN. THE LOCATION WILL THEN BE READ
16913 ;AND A MISS SHOULD BE LOGGED IN THE HIT/MISS REGISTER. THIS READ WILL ALSO
16914 ;BRING VALID DATA INTO CACHE FROM MEMORY. THE MMU WILL AGAIN BE ENABLED
16915 ;WITH BIT <15> SET AND A READ CYCLE DONE. THIS SHOULD INVALIDATE THE CACHE.
16916 ;NEXT THE MMU WILL BE DISABLED AND THE LOCATION READ FOR A THIRD TIME. THIS
16917 ;WILL BE DONE WITH BIT<15> STILL SET. A MISS SHOULD ALSO BE LOGGED. LASTLY
16918 ;BIT<15> WILL BE CLEARED AND THE MMU ENABLED AND THE LOCATION AGAIN READ.
16919 ;THIS TIME A CACHE HIT SHOULD BE LOGGED.
16920 ;
16921 ;
16922 ;BGNTST
16923 ;SETUP MMU REGISTERS
16924 ;WRITE TEST LOCATION WITH PATTERN 1
16925 ;SET BIT<15> IN PDR FOR TEST LOCATION
16926 ;ENABLE MMU
16927 ;WRITE TEST LOCATION WITH PATTERN 2
16928 ;DISABLE MMU AND CLEAR BIT<15> IN PDR
16929 ;READ TEST LOCATION
16930 ;IF HIT/MISS REGISTER BIT<1> SET THEN
16931 ;. ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16932 ;ENDIF
16933 ;SET BIT<15> IN PDR FOR TEST LOCATION
16934 ;ENABLE MMU
16935 ;READ TEST LOCATION (SHOULD INVALIDATE CACHE)
16936 ;DISABLE MMU
16937 ;READ TEST LOCATION
16938 ;IF HIT/MISS REGISTER BIT<1> SET THEN
16939 ;. ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16940 ;ENDIF
16941 ;CLEAR BIT<15> IN PDR
16942 ;TURN ON MMU
16943 ;READ TEST LOCATION
16944 ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
16945 ;. ERROR IN RECORDING HITS IN HIT/MISS
16946 ;ENDIF
16947 ;TURN OFF MMU
16948 ;ENDTST
16949 ;
16950 ;*****
16951 106126 000004 TST10: SCOPE
16952 106130 000240 NOP
16953 106132 005737 003032 TST CCHPAS ;have done enough inclusive passes?
16954 106136 001002 BNE 99$ ; not yet
16955 106140 000240 NOP ; debug aid
16956 106142 000512 BR TST11 ;;GO TO NEXT TEST
16957 106144 000240 99$: NOP
16958 106146 004737 136574 JSR PC, INITMM ;SETUP MEMORY MANAGEMENT REGISTERS
16959 106152 005037 003162 CLR @TSTLOC ;WRITE TEST LOCATION WITH PATTERN 1
16960 106156 012737 177777 001124 MOV #177777,#GDDAT ;SAVE DATA IN MEMORY
16960 106164 052737 100000 172300 BIS #BIT15, KIPDRO ;SET BIT15 IN PDR FOR TEST LOCATION

```

TEST - PDR BIT15 (BYPASS) TEST

16961	106172	005237	177572			INC	SRO	;TURN ON MEMORY MANAGEMENT
16962	106176	012737	177777	003162		MOV	#177777,#TSTLOC	;WRITE TEST LOCATION WITH PATTERN 2
16963	106204	005037	177572			CLR	SRO	;TURN OFF MEMORY MANAGEMENT
16964	106210	042737	100000	172300		BIC	#BIT15, KIPDR0	;CLEAR BIT15 IN PDR FOR TEST LOCATION
16965	106216	042737	001000	177520		BIC	#1000,BCSR	;DISABLE HALT ON BREAK
16966	106224	013701	003162			MOV	#TSTLOC,R1	;READ TEST LOCATION
16967	106230	013737	177752	114150		MOV	HITMIS,RECDAT	;SAVE HIT/MISS REGISTER
16968	106236	032737	000004	114150		BIT	#BIT02,RECDAT	;IF HIT/MISS REGISTER BIT 1 SET
16969	106244	001401				BEQ	2\$;THEN
16970	106246	104023				ERROR	+23	;CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16971	106250	052737	100000	172300	2\$:	BIS	#BIT15, KIPDR0	;SET BIT15 IN PDR FOR TEST LOCATION
16972	106256	005237	177572			INC	SRO	;TURN ON MEMORY MANAGEMENT
16973	106262	005737	003162			TST	TSTLOC	;READ TEST LOCATION
16974	106266	042737	100000	172300		BIC	#BIT15, KIPDR0	;CLEAR BIT 15 IN PDR
16975	106274	005037	177572			CLR	SRO	;TURN OFF MMU
16976	106300	013701	003162			MOV	#TSTLOC,R1	;READ TEST LOCATION
16977	106304	013737	177752	114150		MOV	HITMIS,RECDAT	;STORE HIT/MISS TO REGISTER 2
16978	106312	032737	000004	114150		BIT	#BIT02,RECDAT	;IF HIT/MISS REGISTER BIT 1 SET
16979	106320	001401				BEQ	3\$;THEN
16980	106322	104023				ERROR	+23	;CONDITIONAL BYPASS DOESN'T INVALIDATE O CACHE
16981	106324	005237	177572		3\$:	INC	SRO	;TURN ON MEMORY MANAGEMENT
16982	106330	005737	003162			TST	#TSTLOC	;READ TEST LOCATION ONE MORE TIME
16983	106334	013737	177752	114150		MOV	HITMIS,RECDAT	;STORE HIT/MISS TO REGISTER 2
16984	106342	032737	000004	114150		BIT	#BIT02,RECDAT	;IF HIT/MISS REGISTER BIT 1 NOT SET
16985	106350	001001				BNE	4\$;THEN
16986	106352	104045				ERROR	+45	;ERROR IN RECORDING HITS IN HIT/MISS
16987	106354	042737	000001	177572	4\$:	BIC	#BIT00, SRO	;TURN OFF MMU
16988	106362	052737	001000	177520		BIS	#1000,BCSR	;ENABLE HALT ON BREAK
16989								

TEST - FLUSH CACHE TEST

```

16991 .SBTTL TEST - FLUSH CACHE TEST
16992 ;FLUSH CACHE TEST - THIS TEST WILL VERIFY THAT WHEN CCR BIT<8> IS
16993 ;SET, THE ENTIRE CACHE IS INVALIDATED. FIRST 8K BYTES OF ADDRESSES
16994 ;WILL BE READ TO ALLOCATE CACHE. THEN THE CACHE WILL BE FLUSHED.
16995 ;THE SAME SET OF ADDRESS WILL THEN BE READ AND THE HIT/MISS REGISTER
16996 ;CHECKED AFTER EACH READ FOR A MISS TO BE LOGGED.
16997 ;
16998 ;BGNTST
16999 ;GET FIRST ADDRESS OF 8KB BUFFER
17000 ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
17001 ;DO UNTIL LOOP COUNTER = 0
17002 ;. READ @ADDRESS+
17003 ;ENDDO
17004 ;GET FIRST ADDRESS OF 8KB BUFFER
17005 ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
17006 ;INITIALIZE MISS COUNT TO 4KWORD COUNT
17007 ;FLUSH CACHE
17008 ;DO UNTIL LOOP COUNTER = 0
17009 ;. READ @ADDRESS+
17010 ;. INCREMENT ADDRESS TO READ EVERY 2WORDS
17011 ;. IF HIT/MISS REGISTER BIT<1> SET THEN
17012 ;. DECREMENT MISS COUNT
17013 ;. ENDF
17014 ;ENDDO
17015 ;GET FIRST ADDRESS OF 8KB BUFFER
17016 ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
17017 ;DO UNTIL LOOP COUNTER = 0
17018 ;. READ @ADDRESS+
17019 ;ENDDO
17020 ;GET LAST ADDRESS OF 8KB BUFFER
17021 ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
17022 ;FLUSH CACHE
17023 ;DO UNTIL LOOP COUNTER = 0
17024 ;. READ @ADDRESS-
17025 ;. DECREMENT ADDRESS TO READ EVERY 2WORDS
17026 ;. IF HIT/MISS REGISTER BIT<1> SET THEN
17027 ;. DECREMENT MISS COUNT
17028 ;. ENDF
17029 ;ENDDO
17030 ;IF MISS COUNT NOT EQUAL TO 4096. THEN
17031 ;. ERROR HITS RECORDED AFTER FLUSHING CACHE
17032 ;ENDIF
17033 ;ENDTST
17034 ;
17035 ;*****
TST11: SCOPE
          NOP
17036 106372 000240          TST CCHPAS ;have done enough inclusive passes?
17037 106374 005737 003032          BNE 99$ ; not yet
17038 106400 001002          NOP ; debug aid
17039 106402 000240          BR TST12 ;:GO TO NEXT TEST
17040 106404 000517
17041 106406 000240          99$: NOP
17042 106410 004737 136574          JSR PC, INITMM ;INITIALIZE MMU
17043 106414 012737 001600 172354          MOV #1600,KIPAR6 ;LET PAR6 POINT TO LOW 28K
17044 106422 012701 140000          MOV #140000,R1 ;GET ADDRESS OF 8K BYTE BUFFER
17045 106426 005237 177572          INC SRO ;ENABLE MMU
17046 106432 012702 010000          MOV #4096., R2 ;INIT LOOP COUNTER TO 4K WORD COUNT

```


TEST - UNCONDITIONAL BYPASS TEST

```

17085 .SBTTL TEST - UNCONDITIONAL BYPASS TEST
17086 ;UNCONDITIONAL BYPASS TEST - THIS TEST WILL VERIFY THAT WHEN CLR
17087 ;BIT<9> IS SET, A MEMORY REFERENCE IS FORCED TO MAIN MEMORY. THIS
17088 ;WILL ALSO VERIFY THAT READ AND WRITE HIT INVALIDATE THE CACHE
17089 ;LOCATIONS WHEN BYPASS IS SET.
17090 ;
17091 ;
17092 ;BGNTST
17093 ;
17094 ;READ TEST LOCATIONS TO SETUP POSSIBILITY OF HIT
17095 ;SET CCR BIT<9>
17096 ;
17097 ;READ FIRST TEST LOCATION
17098 ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
17099 ;. ERROR IN RECORDING HITS IN HIT/MISS
17100 ;ENDIF
17101 ;
17102 ;WRITE SECOND TEST LOCATION
17103 ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
17104 ;. ERROR IN RECORDING HITS IN HIT/MISS
17105 ;ENDIF
17106 ;CLEAR CCR BIT<9>
17107 ;
17108 ;READ FIRST LOCATION
17109 ;IF HIT/MISS REGISTER BIT<1> SET THEN
17110 ;. ERROR BYPASS DOESN'T INVALIDATE CACHE
17111 ;ENDIF
17112 ;
17113 ;READ SECOND LOCATION
17114 ;IF HIT/MISS REGISTER BIT<1> SET THEN
17115 ;. ERROR BYPASS DOESN'T INVALIDATE CACHE
17116 ;ENDIF
17117 ;
17118 ;ENDTST
17119 ;
17120 ;
17121 ;
17122 ;*****
TST12: SCOPE
      NOP
17123 106644 000004      TST      CCHPAS          ;have done enough inclusive passes?
17124 106646 000240      BNE      99$           ; not yet
17125 106650 005737 003032      NOP          ; debug aid
17126 106654 001002      BR       TST13        ;;GO TO NEXT TEST
17127 106660 000471      99$:      NOP
17128 106662 000240
17129
17130 106664 005737 003162      TST      @TSTLOC        ;ALLOCATE FIRST TEST LOCATION
17131 106670 005737 003164      TST      @TSTLOC+2      ;ALLOCATE SECOND TEST LOCATION
17132 106674 052737 001000 177746      BIS      @BIT09, CCR    ;SET CCR BIT 9 (CACHE BYPASS)
17133 106702 042737 001000 177520 1$:      BIC      @1000,BCSR     ;DISABLE HALT ON BREAK
17134 106710 005737 003162      TST      @TSTLOC        ;READ FIRST TEST LOCATION
17135 106714 013737 177752 114150      MOV      HITMIS,RECDAT  ;STORE HIT/MISS TO REGISTER 2
17136 106722 032737 000004 114150      BIT      @BIT02,RECDAT  ;IF HIT/MISS REG. BIT 1 NOT SET
17137 106730 001001      BNE      2$           ;THEN
17138 106732 104045      ERROR   +45          ;ERROR IN RECORDING HITS IN HIT/MISS WITH CCR<9>=1
17139 106734 005037 003164 2$:      CLR      @TSTLOC+2      ;WRITE SECOND LOCATION
17140 106740 013737 177752 114150      MOV      HITMIS,RECDAT  ;STORE HIT/MISS TO REGISTER 2

```


TEST - WRITE WRONG DATA PARITY TEST

```

17158 .SBTTL TEST - WRITE WRONG DATA PARITY TEST
17159 ;WRITE WRONG DATA PARITY TEST THIS TEST WILL VERIFY THAT WHEN CCR
17160 ;BIT<6> = 1 AND CCR BITS<7,0> = 0,1, A READ MISS OCCURS AFTER A
17161 ;WRITE. THE WRITE WITH CCR BIT<6> = 1 TO A LOCATION WILL CAUSE A
17162 ;CACHE UPDATE AND WRONG PARITY TO BE WRITTEN SO WHEN THE LOCATION
17163 ;IS READ INSTEAD OF A HIT BEING RECORDED THE CACHE PARITY ERROR
17164 ;WILL CAUSE A CACHE MISS. THIS TEST WILL BE DONE A BYTE AT A TIME.
17165 ;
17166 ;BGNTST
17167 ;SAVE CONTENTS OF VECTOR 114
17168 ;LET VECTOR 114 POINT TO ABORT ROUTINE
17169 ;CLEAR MSER
17170 ;IF MSER NOT CLEAR THEN
17171 ;. ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
17172 ;ENDIF
17173 ;WRITE TEST LOCATION
17174 ;SET BITS<6,0> IN CCR
17175 ;WRITE TEST LOCATION LOW BYTE
17176 ;CLEAR CCR BIT<6> (WRITE WRONG DATA PARITY)
17177 ;INITIALIZE ERROR INDICATORS
17178 ;READ TEST LOCATION LOW BYTE
17179 ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
17180 ;. PARITY ERROR DOESN'T CAUSE MISS
17181 ;ELSE
17182 ;. IF MSER BITS <7:5> ZERO THEN
17183 ;. PARITY ERROR DOESN'T SET MSER PROPERLY
17184 ;. ELSE
17185 ;. SET ERROR IN LOW BYTE INDICATOR
17186 ;. ENDIF
17187 ;. ENDIF
17188 ;ENDIF
17189 ;CLEAR MSER
17190 ;IF MSER NOT CLEAR THEN
17191 ;. ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
17192 ;ENDIF
17193 ;SET CCR BIT<6>
17194 ;WRITE TEST LOCATION HIGH BYTE
17195 ;CLEAR CCR BIT<6>
17196 ;READ TEST LOCATION HIGH BYTE
17197 ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
17198 ;. IF MSER BITS<7:5> NOT SET THEN
17199 ;. PARITY ERROR DOESN'T CAUSE A MISS
17200 ;. ELSE
17201 ;. SET ERROR IN HIGH BYTE INDICATOR
17202 ;. ENDIF
17203 ;ELSE
17204 ;. IF MSER BITS <7:5> ZERO THEN
17205 ;. PARITY ERROR DOESN'T SET MSER
17206 ;. ENDIF
17207 ;ENDIF
17208 ;RESTORE CONTENTS OF VECTOR 114
17209 ;IF ERROR INDICATORS SET THEN
17210 ;. IF ERROR IN BOTH BYTES THEN
17211 ;. PARITY ERROR IGNORED
17212 ;. ELSE
17213 ;. IF ERROR IN LOW BYTE THEN
17214 ;. LOW BYTE PARITY ERROR IGNORED

```

TEST WRITE WRONG DATA PARITY TEST

```

17215      .           ELSE
17216      .           .           HIGH BYTE PARITY ERROR IGNORED
17217      .           .           ENDIF
17218      .           .           ENDIF
17219      .           .           ENDIF
17220      .           .           ;EXIT TST
17221      .           .           ;DATA PARITY ABORT ROUTINE:
17222      .           .           .           ILLEGAL PARITY INTERRUPT
17223      .           .           .           RETURN
17224      .           .           .           ;
17225      .           .           .           ;
17226      .           .           .           ;
17227      .           .           .           ;ENDTST
;*****
TST13:  SCOPE
        NOP
17228 107044 000004      TST CCHPAS ;have done enough inclusive passes?
17229 107050 005737 003032  BNE 99$ ; not yet
17230 107054 001002      NOP ; debug aid
17231 107056 000240      BR TST14 ;:GO TO NEXT TEST
17232 107060 000537
17233 107062 000240 99$:  NOP
17234 107064 013737 000114 003012  MOV @#114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
17235 107072 012737 107350 000114  MOV @DAPAB0,@#114 ;LET VECTOR POINT TO ABORT ROUTINE
17236 107100 005037 177744      CLR MSER ;CLEAR MSER
17237 107104 005737 177744      TST MSER ;IF MSER NOT CLEAR
17238 107110 001401      BEQ 1$ ;THEN
17239 107112 104026      ERROR +26 ;MSER DOES NOT CLEAR ON WRITE REFERENCE
17240 107114 005037 003162 1$:  CLR @#TSTLOC ;WRITE TEST LOCATION TO ALLOCATE CACHE
17241 107120 042737 001000 177520  BIC #1000,BCSR ;DISABLE HALT ON BREAK
17242 107126 012737 000101 177746  MOV #101, CCR ;SET BITS<6,0> IN CCR
17243 107134 112737 000377 003162  MOVB #377, TSTLOC ;WRITE LOW BYTE WITH BAD PARITY
17244 107142 042737 000100 177746  BIC #BIT06, CCR ;CLEAR WRITE WRONG DATA PARITY BIT
17245 107150 005002      CLR R2 ;CLEAR ERROR INDICATORS
17246 107152 105737 003162      TSTB @#TSTLOC ;READ LOW BYTE OF TEST LOCATION
17247 107156 013703 177752      MOV HITMIS, R3 ;SAVE HIT/MISS
17248 107162 032703 000004      BIT #BIT02, R3 ;IF BIT 1 SET IN HIT/MISS REGISTER
17249 107166 001402      BEQ 2$ ;THEN
17250 107170 104027      ERROR +27 ;PARITY ERROR DON'T CAUSE A MISS
17251 107172 000405      BR 3$ ;ELSE
17252 107174 032737 000340 177744 2$:  BIT #340, MSER ;IF MSER BIT<7:5> NOT ZERO
17253 107202 001001      BNE 3$ ;THEN
17254 107204 104030      ERROR +30 ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
17255 107206 005037 177744 3$:  CLR MSER ;CLEAR MSER
17256 107212 005737 177744      TST MSER ;IF MSER NOT CLEAR
17257 107216 001401      BEQ 4$ ;THEN
17258 107220 104026      ERROR +26 ;MSER DOES NOT CLEAR ON WRITE REFERENCE
17259 107222 052737 000100 177746 4$:  BIS #BIT06, CCR ;SET CCR BIT 6 (WRITE WRONG PARITY)
17260 107230 112737 000377 003163  MOVB #377, @#TSTLOC+1 ;WRITE HIGH BYTE OF TEST LOCATION
17261 107236 042737 000100 177746  BIC #BIT06, CCR ;CLEAR WRITE WRONG PARITY BIT
17262 107244 105737 003163      TSTB @#TSTLOC+1 ;READ HIGH BYTE OF TEST LOCATION
17263 107250 013737 177752 114150  MOV HITMIS,RECDAT ;SAVE HIT/MISS
17264 107256 032737 000004 114150  BIT #BIT02,RECDAT ;IF BIT 1 SET IN HIT/MISS REGISTER
17265 107264 001402      BEQ 5$ ;THEN
17266 107266 104027      ERROR +27 ;PARITY ERROR DON'T CAUSE A MISS
17267 107270 000405      BR 6$ ;ELSE
17268 107272 032737 000340 177744 5$:  BIT #340, MSER ;IF BITS <7:5> ZERO
17269 107300 001001      BNE 6$ ;THEN
17270 107302 104030      ERROR +30 ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0

```

TEST - WRITE WRONG DATA PARITY TEST

```

17271 107304 005037 177746      6#: CLR      CCR          ;CLEAR CCR BEFORE EXIT
17272 107310 013737 003012 000114  MOV     SLOC00, 6#114 ;RESTORE CONTENTS OF VECTOR 114
17273 107316 032702 000003      BIT     #3,      R2    ;IF ERROR INDICATORS SET
17274 107322 001401      BEQ     7#          ;THEN
17275 107324 104031      ERROR  +31        ;PARITY ERROR IGNORED
17276 107326 005037 177746      7#: CLR      CCR          ;CLEAR CCR
17277 107332 013737 003012 000114  MOV     SLOC00, 0#114 ;RESTORE PARITY TRAP
17278 107340 052737 001000 177520  BIS     #1000, BC>R   ;ENABLE HALT ON BREAK
17279 107346 000404      BR      TST14       ;;GO TO NEXT TEST
17280
17281
17282 107350 011637 001122      DAPABO: MOV    (SP),  #BDADR ;SAVE ADDRESS THAT CAUSED ABORT
17283 107354 104007      ERROR  +7          ;ILLEGAL PARITY INTERRUPT
17284 107356 000002      RTI
17285

```

TEST WRITE WRONG TAG PARITY

```

17287 .SBTTL TEST WRITE WRONG TAG PARITY
17288 ;WRITE WRONG TAG PARITY THIS TEST WILL VERIFY THAT A READ MISS
17289 ;OCCURS AFTER A WRITE WILL CCR<10> = 1 AND CCR<7,0> = 0,1. THE WRITE
17290 ;TO THE LOCATION WILL CAUSE A CACHE UPDATE BUT THE TAG WILL BE
17291 ;WRITTEN WITH THE WRONG PARITY. WHEN THE LOCATION IS READ INSTEAD
17292 ;OF A CACHE HIT OCCURRING THE PARITY ERROR SHOULD CAUSE A CACHE
17293 ;MISS.
17294 ;
17295 ;BGNTST
17296 ;SAVE CONTENTS OF VECTOR 114
17297 ;LET VECTOR 114 POINT TO ABORT ROUTINE
17298 ;CLEAR MSER
17299 ;SET WRITE WRONG TAG PARITY BIT<10>
17300 ;WRITE TEST LOCATION
17301 ;CLEAR CCR BIT<10> AND SET ABORT DISABLE BIT<0>
17302 ;READ TEST LOCATION
17303 ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
17304 ;. IF MSER BITS <7:5> SET THEN
17305 ;. PARITY ERROR DOESN'T CAUSE MISS
17306 ;. ELSE
17307 ;. PARITY ERROR DOESN'T SET MSER WITH CCR<7>=0
17308 ;. ENDF
17309 ;ENDIF
17310 ;SET WRITE WRONG TAG PARITY BIT<10>
17311 ;WRITE TO A BYTE OF A TEST LOCATION
17312 ;SAVE HIT/MISS REGISTER
17313 ;CLEAR WRITE WRONG TAG PARITY BIT
17314 ;IF BIT<1> NOT SET IN HIT/MISS REGISTER THEN
17315 ;. ERROR
17316 ;ENDIF
17317 ;RESTORE VECTOR 114
17318 ;EXIT TST
17319 ;
17320 ;TAG PARITY ABORT ROUTINE:
17321 ; ILLEGAL PARITY INTERRUPT
17322 ; RETURN
17323 ;ENDTST
17324 ;*****
17325 107360 000004 TST14: SCOPE
17326 107362 000240 NOP
17327 107364 005737 003032 TST CCHPAS ;have done enough inclusive passes?
17328 107370 001002 BNE 99$ ; not yet
17329 107374 000453 BR TST15 ; debug aid
17330 107376 000240 99$: NOP ;;GO TO NEXT TEST
17331 107400 013737 000114 003012 MOV @114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
17332 107406 012737 107514 000114 MOV @TAPAB0,@114 ;LET VECTOR POINT TO ABORT ROUTINE
17333 107414 042737 001000 177520 BIC @1000,BCSR ;DISABLE HALT ON BREAK
17334 107422 005037 177744 CLR MSER ;CLEAR MSER
17335 107426 012737 002000 177746 MOV @BIT10, CCR ;SET WRITE WRONG TAG PARITY
17336 107434 005037 003162 CLR @TSTLOC ;WRITE LOCATION WITH BAD TAG PARITY
17337 107440 012737 000001 177746 MOV @BIT00, CCR ;CLEAR BIT 10 AND SET BIT 0
17338 107446 005737 003162 TST @TSTLOC ;READ TEST LOCATION
17339 107452 013737 177752 114150 MOV HITMIS,RECDAT ;SAVE HIT/MISS REGISTER
17340 107460 032737 000004 114150 BIT @BIT02,RECDAT ;IF BIT 1 SET IN HIT/MISS REGISTER
17341 107466 001401 BEQ 2$ ;THEN
17342 107470 104027 ERROR +27 ;PARITY ERRGR DON'T CAUSE A MISS

```


TEST PARITY ABORT TEST

```

17354 .SBTTL TEST - PARITY ABORT TEST
17355 ;PARITY ABORT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
17356 ;1,0, AN ABORT OCCURS ON THE EXECUTION OF AN INSTRUCTION THAT HAS
17357 ;BEEN WRITTEN WITH WRONG PARITY.
17358 ;
17359 ;
17360 ;BGNTST
17361 ;SAVE VECTOR 114
17362 ;SAVE VECTOR 4
17363 ;LET VECTOR 114 POINT TO ABORT ROUTINE
17364 ;LET VECTOR 4 POINT TO ERROR ROUTINE
17365 ;CLEAR EXPECTING ABORT FLAG
17366 ;SET CCR<10> WRITE WRONG TAG PARITY BIT
17367 ;WRITE TEST ADDRESS
17368 ;CLEAR CCR<10>
17369 ;SET CCR TO #200 ENABLE PARITY ABORTS
17370 ;SET EXPECTING ABORT FLAG
17371 ;READ TEST ADDRESS
17372 ;IF ABORT FLAG NE 0 THEN
17373 ;. ERROR IN PARITY ABORT LOGIC
17374 ;ENDIF
17375 ;RESTORE VECTOR 114
17376 ;RESTORE VECTOR 4
17377 ;EXIT TST
17378 ;
17379 ;ABORT ROUTINE: IF EXPECTING ABORT FLAG NOT SET THEN
17380 ;. ERROR NO ABORT SHOULD HAVE OCCURRED
17381 ;. ELSE
17382 ;. CLEAR (EXPECTING) ABORT FLAG
17383 ;. ENDIF
17384 ;. IF MSER NOT EQUAL TO 100040 THEN
17385 ;. PARITY ABORT LOGIC DOESN'T SET MSER PROPERLY
17386 ;. ENDIF
17387 ;. IF PC = UPDATED PC THEN
17388 ;. ILLEGAL PARITY ABORT
17389 ;. ENDIF
17390 ;. RETURN
17391 ;
17392 ;ENDTST
17393 ;*****
TST15: SCOPE
17394 107524 000004 NOP
17395 107526 000240 TST CCHPAS ;have done enough inclusive passes?
17396 107530 005737 003032 BNE 99$ ; not yet
17397 107534 001002 NOP ; debug aid
17398 107536 000240 BR TST16 ;;GO TO NEXT TEST
17399 107540 000504 99$: NOP
17400 107542 000240
17401 107544 013737 000114 003012 MOV #114, SLOC00 ;SAVE VECTOR 114
17402 107552 013737 000004 003014 MOV #4, SLOC01 ;SAVE VECTOR 4
17403 107560 012737 107676 000114 MOV #ABORTR,#114 ;LET VECTOR 114 POINT TO ABORT ROUTINE
17404 ;
17405 ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCTIONS
17406 ;
17407 107566 012704 107566 MOV #,R4 ;START WITH CURRENT
17408 107572 005724 1$: TST (R4)+ ;READ A WORD
17409 107574 022704 107676 CMP #ABORTR,R4 ;GOT TO ABORT ROUTINE?

```

TEST - PARITY ABORT TEST

```

17410 107600 001374          BNE      1$          ;IF NOT, KEEP ON ALLOCATING
17411 107602 005000          CLR      R0          ;CLEAR EXPECTING ABORT FLAG
17412 107604 042737 001000 177520 BIC      #1000,BCSR ;DISABLE HALT ON BREAK
17413 107612 012737 002000 177746 MOV      #BIT10,CCR ;SET WRITE WRONG PARITY BIT
17414 107620 005037 003162          CLR      @TSTLOC    ;WRITE TEST LOCATION WITH BAD PARITY
17415 107624 012737 000200 177746 MOV      #BIT07,CCR ;ENABLE ABORTS, CLEAR WTMP BIT
17416 107632 005100          COM      R0          ;SET EXPECTING ABORT FLAG
17417 107634 005737 003162 ABORTI: TST      @TSTLOC ;READ TEST LOCATION (SHOULD CAUSE ABORT)
17418 107640 005700          TST      R0          ;IF ABORT FLAG NOT EQUAL ZERO
17419 107642 001401          BEQ      1$          ;THEN
17420 107644 104034          ERROR    +34         ;PARITY ABORT LOGIC DOESN'T WORK
17421 107646 052737 001000 177520 1$: BIS      #1000,BCSR ;ENABLE HALT ON BREAK
17422 107654 013737 003012 000114 MOV      SLOC00, @#114 ;RESTORE VECTOR 114
17423 107662 013737 003014 000004 MOV      SLOC01, @#4  ;RESTORE VECTORE 4
17424 107670 005037 177744          CLR      MSER
17425 107674 000426          BR       TST16      ;;GO TO NEXT TEST
17426
17427
17428 107676 013703 177744 ABORTR: MOV      MSER,R3 ;SAVE MSER
17429 107702 005700          TST      R0          ;IF EXPECTING ABORT FLAG NOT SET
17430 107704 001004          BNE      1$          ;THEN
17431 107706 011637 001122 MOV      (SP), $BDADR ;SAVE ABORT ADDRESS
17432 107712 104007          ERROR    +7         ;ILLEGAL PARITY INTERRUPT
17433 107714 000401          BR       2$          ;ELSE
17434 107716 005000          1$: CLR      R0          ;CLEAR (EXPECTING) ABORT FLAG
17435 107720 022737 100040 177744 2$: CMP      #100040,MSER ;IF MSER NOT EQUAL TO 100040
17436 107726 001404          BEQ      3$          ;THEN
17437 107730 012737 100040 001124 MOV      #100040,$GDDAT ;SAVE PROPER MSER SETTING
17438 107736 104035          ERROR    +35         ;PARITY ABORT DON'T SET MSER PROPERLY
17439 107740 021627 107640 3$: CMP      (SP), $ABORTI+4 ;IF PC EQUAL TO UPDATE PC
17440 107744 001401          BEQ      4$          ;THEN
17441 107746 104007          ERROR    +7         ;ILLEGAL PARITY INTERRUPT
17442 107750 000002          4$: RTI          ;RETURN
17443

```

TEST - PARITY INTERRUPT TEST

```

17445 .SBTTL TEST - PARITY INTERRUPT TEST
17446 ;PARITY INTERRUPT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
17447 ;0,0, A PARITY INTERRUPT OCCURS AFTER EXECUTION OF AN INSTRUCTION
17448 ;THAT HAS BEEN WRITTEN WITH WRONG PARITY.
17449 ;
17450 ;BGNTST
17451 ;SAVE CONTENTS OF 114
17452 ;SETUP VECTOR 114 TO POINT TO INTERRUPT ROUTINE
17453 ;CLEAR EXPECTING INTERRUPT FLAG
17454 ;WRITE TEST ADDRESS WITH BAD PARITY
17455 ;SET EXPECTING INTERRUPT FLAG
17456 ;READ TEST ADDRESS
17457 ;IF INTERRUPT FLAG NE 0 THEN
17458 ;. PARITY INTERRUPT LOGIC DOESN'T WORK
17459 ;ENDIF
17460 ;RESTORE CONTENTS OF VECTOR 114
17461 ;EXIT TST
17462 ;
17463 ;INTERRUPT ROUTINE: IF EXPECTING INTERRUPT FLAG NE 1 THEN
17464 ;. ERROR NO INTERRUPT SHOULD HAVE OCCURRED
17465 ;. ELSE
17466 ;. CLEAR (EXPECTING) INTERRUPT FLAG
17467 ;. ENDIF
17468 ;IF SAVED PC NE TO UPDATED PC THEN
17469 ;. IF PC = TEST INSTRUCTION PC THEN
17470 ;. ERROR INSTRUCTION ABORTED
17471 ;. ELSE
17472 ;. ILLEGAL PARITY ABORT
17473 ;. ENDIF
17474 ;ENDIF
17475 ;IF MSER NE #340 THEN
17476 ;. PARITY INTERRUPT DOESN'T SET MSER PROPERLY
17477 ;. ENDIF
17478 ;RETURN
17479 ;
17480 ;ENDTST
17481 ;:*****
17482 107752 000004 TST16: SCOPE
17483 107754 000240 NOP
17484 107756 005737 003032 TST CCHPAS ;have done enough inclusive passes?
17485 107762 001002 BNE 99$ ; not yet
17486 107764 000240 NOP ; debug aid
17487 107766 000473 BR TST17 ;;GO TO NEXT TEST
17488 107770 000240 99$: NOP
17489 107772 013737 000114 003012 MOV #114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
17490 110000 012737 110106 000114 MOV #INTERR,#114 ;LET VECTOR POINT TO INTERRUPT ROUTINE
17491 ;
17492 ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCTIONS
17493 ;
17494 110006 012704 110006 MOV #.,R4 ;START WITH CURRENT
17495 110012 005724 1$: TST (R4). ;READ A WORD
17496 110014 022704 110106 CMP #INTERR,R4 ;GOT TO INTERRUPT ROUTINE?
17497 110020 001374 BNE 1$ ;IF NOT, KEEP ON ALLOCATING
17498 110022 005001 CLR R1 ;CLEAR EXPECTING INTERRUPT FLAG
17499 110024 042737 001000 177520 BIC #1000,BCSR ;DISABLE HALT ON BREAK
17500 110032 052737 000100 177746 BIS #BIT06,CCR ;SET WRITE WRONG DATA PARITY

```


TEST - PARITY INTERRUPT TEST

```

17501 110040 005037 003162          CLR    @TSTLOC          ;WRITE LOCATION WITH BAD DATA PARITY
17502 110044 005037 177746          CLR    CCR              ;CLEAR WRITE WRONG DATA PARITY
17503 110050 005101                  COM    R1              ;SET EXPECTING INTERRUPT FLAG
17504 110052 005737 003162          INTRPC: TST @TSTLOC     ;READ TEST LOCATION
17505 110056 005701                  TST    R1              ;IF INTERRUPT FLAG NOT EQUAL ZERO
17506 110060 001401                  BEQ    1$              ;THEN
17507 110062 104036                  ERROR  +36            ;PARITY INTERRUPT LOGIC DOESN'T WORK
17508 110064 052737 001000 177520 1$: BIS    @1000,BCSR     ;ENABLE HALT ON BREAK
17509 110072 013737 003012 000114  MOV    SLOC00, @114   ;RESTORE VECTOR 114
17510 110100 005037 177744          CLR    MSER            ;CLEAR MSER
17511 110104 000424                  BR     TST17          ;;GO TO NEXT TEST
17512
17513
17514 110106 005701          INTERR: TST    R1          ;IF EXPECTING INTERRUPT FLAG NOT SET
17515 110110 001004          BNE    1$              ;THEN
17516 110112 011637 001122          MOV    (SP), $BDADR   ;SAVE INTERRUPT ADDRESS
17517 110116 104007          ERROR  +7            ;ILLEGAL PARITY INTERRUPT
17518 110120 000401          BR     2$              ;ELSE
17519 110122 005001          1$: CLR    R1          ;CLEAR (EXPECTING) INTERRUPT FLAG
17520 110124 021627 110056          2$: CMP    (SP), @INTRPC+4 ;IF SAVED PC NOT EQUAL TO UPDATED PC
17521 110130 001401          BEQ    4$              ;THEN
17522 110132 104007          ERROR  +7            ;ILLEGAL PARITY INTERRUPT
17523 110134 022737 000340 177744 4$: CMP    @340, MSER    ;IF MSER NOT EQUAL TO EXPECTED VALUE
17524 110142 001404          BEQ    5$              ;THEN
17525 110144 012737 000340 001124  MOV    @340, $GDDAT   ;SAVE PROPER MSER
17526 110152 104035          ERROR  +35            ;PARITY INTERRUPT DON'T SET MSER PROPERLY
17527 110154 000002          5$: RTI              ;RETURN
17528

```

TEST MISCELLANEOUS PARITY TEST

```

17530 .SBTTL TEST - MISCELLANEOUS PARITY TEST
17531 ;MISCELLANEOUS PARITY TEST - THIS TEST CHECKS THAT BYPASS CYCLES WITH
17532 ;PARITY ERRORS CAUSE CACHE HIT RESPONSE AND THAT FORCE MISS CYCLES
17533 ;IGNORE PARITY ERRORS.
17534 ;
17535 ;BGNTST
17536 ;WRITE A LOCATION WITH BAD PARITY
17537 ;SET BYPASS IN CCR
17538 ;READ THE LOCATION BACK
17539 ;IF NO HIT OR MSER NOT SET THEN
17540 ;. ERROR
17541 ;ENDIF
17542 ;WRITE A LOCATION WITH BAD PARITY AND SET BYPASS
17543 ;IF MSER SET WHILE READING IT BACK
17544 ;. ERROR
17545 ;ENDIF
17546 ;ENDTST
17547 ;*****
17548 110156 000004 TST17: SCOPE
17549 110160 000240 NOP
17550 110162 005737 003032 TST CCHPAS ;have done enough inclusive passes?
17551 110166 001002 BNE 99$ ; not yet
17552 110170 000240 NOP ; debug aid
17553 110172 000503 BR TST20 ;;GO TO NEXT TEST
17554 110174 000240 99$: NOP
17555 110176 012703 110176 MOV #.,R3 ;START WITH CURRENT INSTRUCTION
17556 110202 005723 10$: TST (R3)+ ;READ A WORD
17557 110204 022703 110354 CMP #4$,R3 ;LAST WORD?
17558 110210 001374 BNE 10$ ;IF NOT, CONTINUE
17559 ;
17560 ; CHECK BYPASS AND BAD PARITY
17561 ;
17562 110212 013737 000114 003012 MOV #0114, SLOC00 ;SAVE PARITY VECTOR
17563 110220 012737 110264 000114 MOV #1$,#0114 ;POINT NEW VECTOR
17564 110226 042737 001000 177520 BIC #1000,BCSR ;DISABLE HALT ON BREAK
17565 110234 052737 002101 177746 BIS #BIT10!BIT06!BIT00,CCR ;DATA AND TAG PAR., NO INT.
17566 110242 005037 003152 CLR #TSTLOC ;WRITE CYCLF
17567 110246 012737 001000 177746 MOV #BIT09,CCR ;SET BYPASS
17568 110254 005737 003162 TST #TSTLOC ;BYPASS WITH WRONG PARITY
17569 110260 104045 ERROR +45 ;ERROR
17570 110262 000407 BR 2$ ;
17571 110264 062706 000004 1$: ADD #4,SP ;ADJUST STACK
17572 110270 032737 000340 177744 BIT #340,MSER ;MSER OK?
17573 110276 001001 BNE 2$ ;IF YES, BRANCH
17574 110300 104042 ERROR +42 ;BYPASS WRONG
17575 ;
17576 ; CHECK FORCE MISS AND BAD PARITY
17577 ;
17578 110302 005037 177744 2$: CLR MSER ;CLEAR MSER
17579 110306 005037 177746 CLR CCR ;CLEAR CCR
17580 110312 012737 110346 000114 MOV #3$,#0114 ;POINT NEW VECTOR
17581 110320 052737 002101 177746 BIS #BIT10!BIT06!BIT00,CCR ;DATA AND TAG PAR., NO INTER
17582 110326 005037 003162 CLR #TSTLOC ;FORCE MISS WITH PARITY
17583 110332 012737 000014 177746 MOV #14,CCR ;FORCE MISS
17584 110340 005737 003162 TST #TSTLOC ;ALLOCATE CACHE
17585 110344 000403 BR 4$

```


TEST - MEMORY SYSTEM ERROR REGISTER TEST

```

17594 .SBTTL TEST - MEMORY SYSTEM ERROR REGISTER TEST
17595 ;MEMORY SYSTEM ERROR REGISTER TEST - THIS TEST WILL VERIFY THE
17596 ;FUNCTIONALITY OF BITS <15> AND <7:5> OF THE MEMORY SYSTEM ERROR
17597 ;REGISTER. THIS TEST WILL USE THE WRITE WRONG PARITY BITS (BITS<10>
17598 ;AND <6> OF THE CCR) TO WRITE BAD PARITY INTO A LOCATION. THE
17599 ;LOCATION WILL THEN BE READ WITH CACHE TRAPS ENABLED AND THE MSER
17600 ;WILL BE CHECKED AFTER THE ABORT FOR THE CORRECT BIT(S) BEING SET.
17601 ;THIS WILL BE DONE FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR WORD
17602 ;ACCESSES THEN REPEATED FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR
17603 ;BYTE ACCESSES. THE TEST WILL THEN BE REPEATED A THIRD TIME FOR BYTE
17604 ;ACCESSES AND ABORTS DISABLED. THE MSER SHOULD CONTAIN BITS <7:5>
17605 ;SET TO 1'S AND BIT <15> A ZERO FOR ALL COMBINATIONS.
17606 ;
17607 ;BGNTST
17608 ;SAVE CONTENTS OF VECTOR 114
17609 ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17610 ;INITIALIZE LOOP COUNTER
17611 ;DO UNTIL ALL WORD COMBINATIONS CHECKED
17612 ;. INITIALIZE MSER
17613 ;. SETUP CCR FROM CCR TABLE
17614 ;. READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17615 ;. CLEAR <10,6> FROM CCR
17616 ;. SETUP CCR FOR ABORT
17617 ;. READ TEST LOCATION ;THIS COULD CAUSE TRAP
17618 ;. IF EXPECTED DATA NE RECEIVED DATA THEN
17619 ;. ERROR IN SETTING MSER
17620 ;.
17621 ;. ENDIF
17622 ;. UPDATE LOOP COUNTER
17623 ;ENDDO
17624 ;INITIALIZE LOOP COUNTER
17625 ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17626 ;. INITIALIZE MSER
17627 ;. SETUP CCR FROM CCR TABLE
17628 ;. READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17629 ;. CLEAR <10,6> FROM CCR
17630 ;. SETUP CCR FOR ABORT
17631 ;. READ LOW BYTE TEST LOCATION
17632 ;. GET EXPECTED BYTE DATA FROM TABLE
17633 ;. IF EXPECTED DATA NE RECEIVED DATA THEN
17634 ;. ERROR IN SETTING MSER
17635 ;.
17636 ;. ENDIF
17637 ;. INITIALIZE MSER
17638 ;. READ HIGH BYTE TEST LOCATION
17639 ;. GET EXPECTED BYTE DATA FROM TABLE
17640 ;. IF EXPECTED DATA NE RECEIVED DATA THEN
17641 ;. ERROR IN SETTING MSER
17642 ;.
17643 ;. ENDIF
17644 ;. INCREMENT LOOP COUNTER
17645 ;ENDDO
17646 ;INITIALIZE LOOP COUNTER
17647 ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17648 ;. INITIALIZE MSER
17649 ;. SETUP CCR FROM CCR TABLE
17650 ;. READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY

```

TEST - MEMORY SYSTEM ERROR REGISTER TEST

```

17651      ;.      IF RECEIVED DATA NE TO #340 THEN
17652      ;.      .      ERROR IN SETTING MSER
17653      ;.      ENDIF
17654      ;.      INITIALIZE MSER
17655      ;.      READ HIGH BYTE TEST LOCATION
17656      ;.      IF RECEIVED DATA NE #340 THEN
17657      ;.      .      ERROR IN SETTING MSER
17658      ;.      ENDIF
17659      ;.      INCREMENT LOOP COUNTER
17660      ;ENDDO
17661      ;EXIT  TST
17662      ;
17663      ;CCR TABLE:      0
17664      ;                  100
17665      ;                  2000
17666      ;                  2100
17667      ;EXPECTED WORD DATA:      0
17668      ;                  100300
17669      ;                  100040
17670      ;                  100340
17671      ;EXPECTED BYTE DATA:      0
17672      ;                  0
17673      ;                  100100
17674      ;                  100200
17675      ;                  100040
17676      ;                  100040
17677      ;                  100140
17678      ;                  100240
17679      ;ABORT ROUTINE: RTI
17680      ;
17681      ;ENDTST
17682      ;*****
17683      110402 000004      TST20:  SCOPE
17684      110404 000240      NOP
17685      110406 005737      TST      CCHPAS      ;have done enough inclusive passes?
17686      110412 001003      BNE      99$      ; not yet
17687      110414 000240      NOP      ; debug aid
17688      110416 000137      JMP      10$      ; yes skip this
17689      110422 000240      99$:  NOP
17690      110424 013737      MOV      @#114, SLOC00      ;SAVE CONTENTS OF VECTOR 114
17691      110432 012737      MOV      @ABROUT,@#114      ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17692      110440 012704      MOV      #4, R4      ;INITIALIZE LOOP COUNTER
17693      110444 012700      MOV      @CCRTBL,R0      ;GET ADDRESS OF CCR TABLE
17694      110450 012701      MOV      @EXPWDT,R1      ;GET ADDRESS OF EXPECTED DATA TABLE
17695      110454 042737      BIC      #1000,BCSR      ;DISABLE HALT ON BREAK
17696      110462 005037      1$:  CLR      MSER      ;INITIALIZE MSER DATA
17697      110466 012037      MOV      (R0)+, CCR      ;SETUP CCR FROM CCR TABLE
17698      110472 005037      CLR      @#TSTLOC      ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17699      110476 012737      MOV      @BIT07, CCR      ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17700      110504 005737      TST      @#TSTLOC      ;READ TEST LOCATION
17701      110510 021137      CMP      (R1), MSER      ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17702      110514 001403      BEQ      2$      ;DATA THEN
17703      110516 011137      MOV      (R1), $GDDAT      ;SAVE PROPER MSER SETTING
17704      110522 104035      ERROR   +35      ;MSER NOT SET PROPERLY
17705      110524 005721      2$:  TST      (R1)+      ;INCREMENT POINTER THRU MSER TABLE
17706      110526 005737      TST      @#TSTLOC+8192.      ;TO INSURE MISS ON THE NEXT LOOP

```


TEST - MEMORY SYSTEM ERROR REGISTER TEST

```

17764 111060 005037 177744          CLR      MSER          ;CLEAR ERROR REGISTER
17765 111064 052737 001000 177520    BIS      @1000,BCSR    ;ENABLE HALT ON BREAK
17766 111072 013737 003012 000114    MOV     SLOCOO, @#114 ;RESTORE VECTOR 114
17767 111100 005037 177746          CLR      CCR          ;CLEAR ALL BIT IN CCR
17768 111104          10$:          BR      TST21          ;;GO TO NEXT TEST
      111104 000421
17769
17770
17771 111106 000000          CCRTBL: .WORD 0
17772 111110 000100          .WORD 100
17773 111112 002000          .WORD 2000
17774 111114 002100          .WORD 2100
17775
17776 111116 000000          EXPWDT: .WORD 0
17777 111120 100300          .WORD 100300
17778 111122 100040          .WORD 100040
17779 111124 100340          .WORD 100340
17780
17781 111126 000000          EXPBDT: .WORD 0
17782 111130 000000          .WORD 0
17783 111132 100100          .WORD 100100
17784 111134 100200          .WORD 100200
17785 111136 100040          .WORD 100040
17786 111140 100040          .WORD 100040
17787 111142 100140          .WORD 100140
17788 111144 100240          .WORD 100240
17789
17790 111146 000002          ABROUT: RTI
17791
17792

```

TEST - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABO

```

17794 .SBTTL TEST CHECK PARITY ABORTS BLOCKED BY NON EXISTENT MEMORY ABORT
17795 ;CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT - THIS TEST WILL
17796 ;VERIFY THAT IF A PARITY ERROR OCCURS ON THE SAME ADDRESS REFERENCE AS A
17797 ;NON-EXISTENT MEMORY ERROR THAT THE CACHE DATA PATH GATE ARRAY BLOCKS THE
17798 ;PARITY ERROR TO THE J 11 CHIP SET. THIS WILL BE DONE BY USING THE DIAGNOSTIC
17799 ;BIT TO CAUSE A PARITY ERROR IN A CACHE REFERENCE THAT DOES NOT HAVE A
17800 ;CORRESPONDING ADDRESS IN MAIN MEMORY. THE ADDRESS WILL THEN BE READ CAUSING
17801 ;BOTH A CACHE PARITY ERROR AND A NON-EXISTENT MEMORY ERROR. TO AVOID HAVING
17802 ;TO SIZE THE ENTIRE MEMORY TO FIND A NON-EXISTENT MEMORY ADDRESS THIS TEST
17803 ;WILL USE THE LARGEST NON-I/O ADDRESS (17757776). THE TEST WILL FIRST READ
17804 ;THIS ADDRESS AND IF A NXM TRAP OCCURS THE TEST WILL BE DONE. IF THE ACCESS
17805 ;TO THIS ADDRESS DOES NOT TRAP THEN THE TEST WILL BE SKIPPED.
17806 ;
17807 ;BGNTST
17808 ;SAVE CONTENTS OF VECTOR 4
17809 ;SAVE CONTENTS OF VECTOR 114
17810 ;LET VECTOR 4 POINT TO CONTINUE TESTING (A:)
17811 ;LET PAR6 = #177400
17812 ;ACCESS ADDRESS 157776 (PHYSICAL 17757776)
17813 ;IF NO TRAP THEN
17814 ;. GOTO ENCTST
17815 ;ENDIF
17816 ;A:
17817 ;SET DIAGNOSTIC AND WRITE WRONG PARITY BITS IN CCR
17818 ;WRITE ADDRESS 157776
17819 ;CLEAR CCR
17820 ;SET PARITY ERROR ABORT BIT IN CCR
17821 ;LET VECTOR 4 POINT TO CONTINUE TESTING (B:)
17822 ;LET VECTOR 114 POINT TO NXM-PARITY ERROR ROUTINE
17823 ;READ ADDRESS 157776
17824 ;IF NO TRAP THEN
17825 ;. ERROR IN ABORT LOGIC
17826 ;ENDIF
17827 ;B:
17828 ;CLEAR CCR
17829 ;RESTORE CONTENTS OF VECTOR 114
17830 ;RESTORE CONTENTS OF VECTOR 4
17831 ;EXIT TST
17832 ;
17833 ;
17834 ;NXM-PARITY ERROR ROUTINE: RESET STACK AFTER TRAP
17835 ; PARITY ABORT NOT BLOCKED BY NXM
17836 ; GOTO B:
17837 ;ENDTST
17838 ;*****
TST21: SCOPE
17839 111150 000004 NOP
17840 111152 000240
17841 111154 005737 003032 TST CCHPAS ;have done enough inclusive passes?
17842 111160 001002 BNE 99# ; not yet
17843 111162 000240 NOP ; debug aid
17844 111164 000475 BR TST22 ;;GO TO NEXT TEST
17845 111166 000240 99#: NOP
17846 111170 013737 000004 003C12 MOV #4, SLOC00 ;SAVE CONTENTS OF VECTOR 4
17847 111176 013737 000114 003C14 MOV #114, SLOC01 ;SAVE CONTENTS OF VECTOR 114
17848 111204 012737 111232 000C04 MOV #11, #4 ;LET VECTOR 4 POINT TO CONTINUE TESTING
17849 111212 012737 177400 172354 MOV #177400,KIPAR6 ;LET PAR6 = OFFSET TO HIGHEST MEMORY

```


TEST - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABO

```

17850 111220 005237 177572      INC      SRO      ;TURN ON MMU
17851 111224 005737 157776      TST      @157776 ;ACCESS ADDRESS 17757776
17852 111230 000431                BR      ABOEXT ;IF NO TRAP SKIP TEST
17853 111232 062706 000004      19:    ADD      @4, SP ;RESET STACK AFTER TRAP
17854 111236 042737 001000 177520      BIC      @1000,BCSR ;DISABLE HALT ON BREAK
17855 111244 012737 000102 177746      MOV      @102, CCR ;SET DIAG. AND WRITE WRONG PARITY BITS
17856 111252 005037 157776      CLR      @157776 ;WRITE TO ADDRESS 17757776
17857 111256 012737 000200 177746      MOV      @200, CCR ;CLEAR CCR AND SET PARITY ABORT BIT
17858 111264 012737 111310 000004      MOV      @29, @4 ;LET VECTOR 4 POINT TO CONTINUE TESTING
17859 111272 012737 111350 000114      MOV      @NXMPAR,@114 ;LET VECTOR 114 POINT TO ERROR ROUTINE
17860 111300 005737 157776      TST      @157776 ;READ ADDRESS 17757776 (SHOULD TRAP)
17861 111304 104037                ERROR   +37 ;NXM AND PARITY ABORT DIN'T HAPPEN
17862 111306 000402                BR      ABOEXT ;GO EXIT TEST
17863 111310 062706 000004      29:    ADD      @4, SP ;RESET STACK AFTER TRAP
17864 111314 005037 177746      ABOEXT: CLR      CCR ;CLEAR CCR FOR EXIT
17865 111320 005037 177572                CLR      SRO ;DISABLE MMU
17866 111324 052737 001000 177520      BIS      @1000,BCSR ;ENABLE HALT ON BREAK
17867 111332 013737 003012 000004      MOV      SLOC00, @4 ;RESTORE VECTOR 4
17868 111340 013737 003014 000114      MOV      SLOC01, @114 ;RESTORE VECTOR 114
17869 111346 000404                BR      TST22 ;GO TO NEXT TEST
17870
17871
17872
17873 111350 062706 000004      NXMPAR: ADD      @4, SP ;RESET STACK AFTER TRAP
17874 111354 104040                ERROR   +40 ;PARITY ABORT NOT BLOCKED BY NXM TRAP
17875 111356 000002                RTI
17876

```

TEST - MULTIPROCESSING INSTRUCTION TESTS

```

17878 .SBTTL TEST - MULTIPROCESSING INSTRUCTION TESTS
17879 ;MULTIPROCESSING INSTRUCTION TESTS - THIS TEST WILL VERIFY THAT THE MULTI-
17880 ;PROCESSING INSTRUCTIONS DO A BYPASS OF THE CACHE. THIS TEST WILL NOT VERIFY
17881 ;THE REST OF THE FUNCTIONALITY OF THESE INSTRUCTIONS BECAUSE THAT WILL ALREADY
17882 ;HAVE BEEN CHECKED IN THE BASE INSTRUCTION TESTS. THE TEST WILL FIRST ALLOCATE
17883 ;AN ADDRESS IN CACHE THEN A TSTSET INSTRUCTION WILL BE DONE AT THAT ADDRESS.
17884 ;A HIT SHOULD BE RECORDED ON THE ACCESS. NEXT, THE ADDRESS WILL BE READ AND
17885 ;A MISS SHOULD BE RECORDED BECAUSE THE FORCED BYPASS ON THE TSTSET INSTRUCTION
17886 ;SHOULD HAVE INVALIDATED THE CACHE ENTRY. THE SAME SEQUENCE WILL THEN BE
17887 ;REPEATED FOR THE WRTLCK INSTRUCTION.
17888 ;
17889 ;BGNTST
17890 ;READ TEST LOCATION TO ALLOCATE CACHE
17891 ;DO TSTSET INSTRUCTION
17892 ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
17893 ;. ERROR IN MULTI-PROCESSOR HOOKS
17894 ;ENDIF
17895 ;READ TEST LOCATION (ALSO ALLOCATES CACHE FOR WRTLCK)
17896 ;DO WRTLCK INSTRUCTION
17897 ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
17898 ;. ERROR IN MULTI-PROCESSOR HOOKS
17899 ;ENDIF
17900 ;READ TEST LOCATION
17901 ;DO ASRB INSTRUCTION
17902 ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
17903 ;. ERROR IN MULTI-PROCESSOR HOOKS
17904 ;ENDIF
17905 ;ENDTST
17906 ;NOTE: THE CODE IS POSITION DEPENDENT
17907
17908 ;:*****
TST22: SCOPE
17909 111360 000004 NOP
17910 111362 000240 TST CCHPAS ;have done enough inclusive passes?
17911 111364 005737 003032 BNF 99$ ; not yet
17912 111370 001002 NOP ; debug aid
17913 111372 000240 BR TST23 ;;GO TO NEXT TEST
17914 111374 000501
17915 111376 000240 99$: NOP
17916 111400 042737 001000 177520 BIC #1000,BCSR ;DISABLE HALT ON BREAK
17917 111406 005737 003162 TST TSTLOC ;READ TEST LOCATION TO ALLOCATE CACHE
17918 ; TSTSET TSTLOC ;DO TSTSET INSTRUCTION
17919 111412 007237 7$: .WORD 7237 ;THESE NEXT TWO LOCATIONS ARE THE TSTSET
17920 111414 003162 .WORD TSTLOC ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
17921 111416 013737 177752 114150 MOV HITMIS,RECDAT ;STORE REGISTER
17922 111424 032737 000010 114150 BIT #BIT3,RECDAT ;IF HIT/MISS REGISTER BIT 3 NOT SET
17923 111432 001001 BNE 1$ ;THEN
17924 111434 104045 ERROR +45 ;ERROR IN RECORDING HITS IN HIT/MISS
17925 111436 032737 000004 114150 1$: BIT #BIT2,RECDAT ;IF HIT/MISS REGISTER BIT 2 SET
17926 111444 001404 BEQ 2$ ;THEN
17927 111446 013737 111412 001126 MOV #7$, $BDDAT ;SAVE OPCODE
17928 111454 104041 ERROR +41 ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MIS
S
17929 ;2$: WRTLCK TSTLOC ;DO WRTLCK INSTRUCTION
17930 111456 000240 2$: NOP ;PUT THE WRITE LOCK CODE ON THE RIGHT $$$
17931 111460 000240 NOP ;BOUNDARY. $$$
17932 111462 007337 .WORD 7337 ;THESE NEXT TWO WORDS ARE THE WRTLCK
17933 111464 003162 .WORD TSTLOC ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY

```

TEST - MULTIPROCESSING INSTRUCTION TESTS

17934	111466	013737	177752	114150	MOV	HITMIS,RECDAT	;STORE REGISTER
17935	111474	032737	000010	114150	BIT	#BIT3,RECDAT	;IF HIT/MISS REGISTER BIT 3 NOT SET
17936	111502	001001			BNE	3#	;THEN
17937	111504	104045			ERROR	+45	;ERROR IN RECORDING HITS IN HIT/MISS
17938	111506	032737	000004	114150	3#:	BIT	#BIT2,RECDAT
17939	111514	001404			BEQ	4#	;IF HIT/MISS REGISTER BIT 2 SET
17940	111516	013737	111456	001126	MOV	#2#, \$BDDAT	;THEN
17941	111524	104041			ERROR	+41	;SAVE OP CODE
							;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MIS
S							
17942	111526	106237	003162		4#:	ASRB	TSTLOC
17943	111532	013737	177752	114150	MOV	HITMIS,RECDAT	;THE THIRD BYPASS INSTRUCTION
17944	111540	032737	000010	114150	BIT	#BIT3,RECDAT	;STORE REGISTER
17945	111546	001001			BNE	5#	;IF HIT/MISS REGISTER BIT 3 NOT SET
17946	111550	104045			ERROR	+45	;THEN
17947	111552	052737	001000	177520	5#:	BIS	#1000,BCSR
17948	111560	032737	000004	114150	BIT	#BIT2,RECDAT	;ERROR IN RECORDING HITS IN HIT/MISS
17949	111566	001404			BEQ	TST23	;ENABLE HALT ON BREAK
17950	111570	013737	111526	001126	MOV	#4#, \$BDDAT	;IF HIT/MISS REGISTER BIT 2 SET
17951	111576	104041			ERROR	+41	;EXIT TEST
							;SAVE OP CODE
							;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MIS
S							
17952							
17953							
17954							

TEST - DATA STORE RAM TESTS

```

17956 .SBTTL TEST - DATA STORE RAM TESTS
17957 ;DATA STORE RAM TESTS - THERE ARE TWO TESTS FOR THE DATA STORE RAM.
17958 ;THE FIRST TEST WILL BE A NO DUAL ADDRESSING TEST AND THE SECOND
17959 ;TEST WILL BE A DATA RELIABILITY TEST. THE NO DUAL ADDRESSING TEST
17960 ;WILL FIRST WRITE EACH WORD OF THE CACHE RAM WITH ITS WORD ADDRESS
17961 ;AND VERIFY THE CONTENTS WITH A READ OF EACH ADDRESS. THEN EACH
17962 ;BYTE WILL BE WRITTEN WITH ITS ADDRESS AND CHECKED. THE DATA
17963 ;RELIABILITY TEST THAT WILL BE USED IS A TEST CALLED MOVING INVERSIONS.
17964 ;
17965 ;BGNTST 1
17966 ;SETUP MMU REGISTERS TO HAVE BYPASS ON KERNAL SPACE AND NO
17967 ; BYPASS ON USER SPACE
17968 ;LET PS EQUAL KERNAL FOR CURRENT MODE AND USER FOR PREVIOUS
17969 ; MODE
17970 ;ENABLE MMU
17971 ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
17972 ;CLEAR DATA TO BE WRITTEN
17973 ;SET DIAGNOSTIC BIT (BIT<1>) IN CCR
17974 ;DO UNTIL DATA TO BE WRITTEN EQUALS 20000(8)
17975 ;. WRITE DATA TO ADDRESS
17976 ;. ADD 2 TO ADDRESS
17977 ;. ADD 2 TO DATA TO BE WRITTEN
17978 ;ENDDO
17979 ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
17980 ;CLEAR EXPECTED DATA
17981 ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
17982 ;. READ ADDRESS
17983 ;. IF RECEIVED DATA NE EXPECTED DATA THEN
17984 ;. GOTO DATA STORE PARITY ERROR ROUTINE
17985 ;. ENDF
17986 ;. ADD 2 TO EXPECTED DATA
17987 ;. ADD 2 TO ADDRESS
17988 ;ENDDO
17989 ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
17990 ;CLEAR DATA TO BE WRITTEN
17991 ;DO UNTIL ALL BYTES CHECKED
17992 ;. WRITE DATA TO ADDRESS
17993 ;. ADD 1 TO ADDRESS
17994 ;. ADD 1 TO DATA TO BE WRITTEN
17995 ;ENDDO
17996 ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
17997 ;CLEAR EXPECTED DATA
17998 ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
17999 ;. READ ADDRESS
18000 ;. IF RECEIVED DATA NE EXPECTED DATA THEN
18001 ;. GOTO DATA STORE PARITY ERROR ROUTINE
18002 ;. ENDF
18003 ;. ADD 1 TO EXPECTED DATA
18004 ;. ADD 1 TO ADDRESS
18005 ;ENDDO
18006 ;ENDTST
18007 ;
18008 ;*****
18009 111600 000004 TST3: SCOPE
18010 111602 000240 NOP
18011 111604 005737 003032 TST CCHPAS ;have done enough inclusive passes?
18012 111610 001002 BNE 99$ ; not yet

```


TEST - DATA STORE RAM TESTS

18069	112122	000747				BR	8\$;ENDDO
18070	112124	052737	001000	177520	10\$:	BIS	Ø1000.BCSR	;ENABLE HALT ON BREAK
18071	112132	005037	177572			CLR	SRO	;TURN OFF MMU
18072								



TEST - TAG STORE RAM TESTS

```

18074 .SBTTL TEST - TAG STORE RAM TESTS
18075 ;TAG STORE RAM TESTS - THERE ARE TWO TESTS FOR THE TAG STORE RAMS. THE FIRST
18076 ;TEST IS AN ADDRESSING TEST TO ENSURE NO DUAL ADDRESSING OF THE TAG STORE.
18077 ;THE SECOND TEST IS A "MOVING INVERSIONS" TEST THAT CHECKS THE DATA RELIABILITY
18078 ;OF THE RAM STORE.
18079 ;
18080 ;DUAL ADDRESSING TEST - THIS WILL BE DONE BY FIRST FLUSHING THE CACHE, THEN
18081 ;THE FIRST 512(10) LOCATIONS (BLOCK 0) WILL BE WRITTEN WITH ADDRESSES THAT
18082 ;WILL CAUSE A INCREMENTING PATTERN TO BE STORED IN THOSE LOCATIONS OF THE TAG
18083 ;STORE RAM. NEXT THE ENTIRE 4K ADDRESS RANGE WILL BE READ. THE HIT/MISS
18084 ;REGISTER SHOULD ONLY REPORT HITS ON THE ADDRESSES THAT WERE ALLOCATED. THIS
18085 ;PROCESS WILL BE REPEATED FOR EACH BLOCK.
18086 ;
18087 ;INITIALIZE LOW ADDRESS
18088 ;SETUP AND ENABLE MMU
18089 ;INITIALIZE BLOCK COUNTER
18090 ;DO UNTIL ALL BLOCKS TESTED (8 TIMES)
18091 ;. FLUSH CACHE AND SET DIAGNOSTIC BIT
18092 ;. LET CURRENT ADDRESS = LOW ADDRESS
18093 ;. INITIALIZE ALLOCATION COUNTER
18094 ;. DO UNTIL ENTIRE BLOCK ALLOCATED (1000 TIMES)
18095 ;. READ CURRENT ADDRESS
18096 ;. ELSE
18097 ;. WRITE CURRENT ADDRESS
18098 ;. ENDIF
18099 ;. UPDATE CURRENT ADDRESS (ALSO ADD 200 TO PAR)
18100 ;. FOR I/O PAGE WRITE THE SAME ADDRESS AS BEFORE
18101 ;. ENDDO
18102 ;. INITIALIZE GOOD ADDRESS
18103 ;. INITIALIZE CURRENT ADDRESS
18104 ;. INITIALIZE LOCATION COUNTER
18105 ;. SAVE CONTENTS OF VECTOR 114
18106 ;. LET VECTOR 114 POINT TO TAG STORE PARITY ABORT ROUTINE
18107 ;. DO UNTIL ALL LOCATIONS CHECKED (1000 TIMES)
18108 ;. INITIALIZE CHECK COUNTER
18109 ;. DO UNTIL ADDRESS IN EACH BLOCK CHECKED
18110 ;. READ CURRENT ADDRESS
18111 ;. IF HIT THEN
18112 ;. IF CURRENT ADDRESS NE GOOD ADDRESS THEN
18113 ;. ERROR
18114 ;. ENDIF
18115 ;. ELSE
18116 ;. IF CURRENT ADDRESS EQUAL GOOD ADDRESS THEN
18117 ;. ERROR
18118 ;. ENDIF
18119 ;. ENDIF
18120 ;. UPDATE GOOD ADDRESS (ADD #2)
18121 ;. UPDATE CURRENT ADDRESS
18122 ;. ENDDO
18123 ;. UPDATE LOW ADDRESS (ADD #2000)
18124 ;. ENDDO
18125 ;ENDDO
18126 ;DISABLE MMU
18127 ;RESTORE VECTOR 114
18128 ;ENDTST
18129 ;
18130 ;:*****

```

TEST - TAG STORE RAM TESTS

18131	112136	000004			TST24:	SCOPE		
18132	112140	000240				NOP		
18133	112142	005737	003032			TST	CCHPAS	;have done enough inclusive passes?
18134	112146	001003				BNE	99\$; not yet
18135	112150	000240				NOP		; debug aid
18136	112152	000137	112710			JMP	18\$; yes skip this
18137	112156	000240			99\$:	NOP		
18138	112160	013737	000004	001160		MOV	004,\$TMP0	;STORE TIMEOUT VECTOR
18139	112166	012737	112712	000004		MOV	020\$,004	;POINT NEW
18140	112174	012737	140000	003016		MOV	0140000,LOWADD	;INITIALIZE LOW ADDRESS (USE PAR6)
18141	112202	004737	136574			JSR	PC, INITMM	;INITIALIZE MMU
18142	112206	005237	177572			INC	SRO	;ENABLE MMU
18143	112212	012737	000020	172516		MOV	0BIT04,MMR3	;ENABLE 22-BIT MAPPING
18144	112220	012737	177770	003154		MOV	0-10, LOOPIN	;DO UNTIL ALL BLOCKS TESTED
18145	112226	012701	000000		1\$:	MOV	00,R1	;DO IN 2 WORDS TO AVOID PMI MEMORY
18146	112232	042737	001000	177520	9\$:	BIC	01000,BCSR	;DISABLE HALT ON BREAK
18147	112240	005037	172354			CLR	KIPAR6	;SET UP PAR6 FOR THIS TEST
18148	112244	012737	000402	177746		MOV	0402, CCR	;FLUSH CACHE AND SET DIAG BIT
18149	112252	013737	003016	114170		MOV	LOWADD, CURADD	;GET FIRST ADDRESS IN CURRENT BLOCK
18150	112260	012737	177400	003152		MOV	0-400, ALLCTR	;DO UNTIL ALL ADDRESSES ALLOCATED
18151	112266	022737	002000	172354	2\$:	CMP	02000, KIPAR6	;IF ADDRESS LESS THAN 32K
18152	112274	002413				BLT	3\$;THEN
18153	112276	052737	100000	172314		BIS	0BIT15,KIPDR6	;SET BYPASS
18154	112304	017702	001660			MOV	0CURADD,R2	;STORE CURRENT DATA
18155	112310	042737	100000	172314		BIC	0BIT15,KIPDR6	;ALLOCATE NEXT ACCESS
18156	112316	010277	001646			MOV	R2,0CURADD	;WRITE ALLOCATE
18157	112322	000402				BR	4\$;ELSE
18158	112324	005077	001640		3\$:	CLR	0CURADD	;WRITE CURRENT ADDRESS
18159	112330	062737	000002	114170	4\$:	ADD	02, CURADD	;UPDATE CURRENT ADDRESS
18160	112336	062737	000200	172354		ADD	0200, KIPAR6	
18161	112344	022737	177600	172354		CMP	0177600,KIPAR6	;REACHED I/O PAGE?
18162	112352	001003				BNE	10\$;BRANCH IF NOT
18163	112354	162737	000200	172354		SUB	0200,KIPAR6	;DON'T UPDATE PAR FOR I/O PAGE
18164	112362	005237	003152		10\$:	INC	ALLCTR	;IF ALL ADDRESSES ALLOCATED
18165	112366	002737				BLT	2\$;ENDDO
18166	112370	052737	000004	177746		BIS	0BIT02, CCR	;RUN WITH FORCE MISS
18167	112376	013737	003016	003020		MOV	LOWADD, GOODAD	;INITIALIZE GOOD ADDRESS
18168	112404	060137	003020			ADD	R1, GOODAD	
18169	112410	012737	140000	114170		MOV	0140000,CURADD	;GET FIRST ADDRESS
18170	112416	060137	114170			ADD	R1, CURADD	;START WITH 1 OR 2 LOCATION
18171	112422	005037	172354			CLR	KIPAR6	
18172	112426	072127	000006			ASH	06,R1	
18173	112432	060137	172354			ADD	R1,KIPAR6	;MAKE SURE ON THE RIGHT BOUNDARY
18174	112436	012737	177600	003152		MOV	0-200, ALLCTR	;DO UNTIL ALL LOCATIONS CHECKED
18175	112444	012737	177770	114144	5\$:	MOV	0-10, DCOUNT	;DO UNTIL ADDRESS IN EACH BLOCK CHECKED
18176	112452	013737	114170	001122	6\$:	MOV	CURADD, 0BDADR	;IN CASE OF ERRORS
18177	112460	042737	160000	001122		BIC	0160000,0BDADR	;CLEAR PAR BITS
18178	112466	042737	000004	177746		BIC	0BIT02, CCR	;CLEAR FORCE MISS
18179	112474	005777	001470			TST	0CURADD	;READ CURRENT ADDRESS
18180	112500	013737	177752	114150		MOV	HITMIS, RECDAT	;STORE REGISTER
18181	112506	032737	000004	114150		BIT	0BIT2, RECDAT	;IF ACCESS WAS A HIT
18182	112514	001406				BEQ	7\$;THEN
18183	112516	023737	114170	003020		CMP	CURADD, GOODAD	;IF CURRENT ADDRESS NOT EQUAL TO GOOD
18184	112524	001407				BEQ	8\$;ADDRESS THEN
18185	112526	104046				ERROR	+46	;ERROR IN TAG STORE
18186	112530	000405				BR	8\$	

TEST - STANDALONE MODE TEST

```

18218 .SBTTL TEST STANDALONE MODE TEST
18219 ;THIS TEST VERIFIES THAT NMX CAN BE CREATED IN STANDALONE MODE.
18220 ;
18221 ;ALLOCATE INSTRUCTIONS IN CACHE
18222 ;GUARANTEE MISS ON TEST LOCATION
18223 ;IN STANALONE MODE ACCESS TEST LOCATION
18224 ;IF NO TIMEOUT THEN
18225 ;. ERROR
18226 ;ENDIF
18227 ;
18228 ;*****
18229 112714 000004 TST25: SCOPE
18230 112716 000240 NOP
18231 112720 005737 003032 TST CCHPAS ;have done enough inclusive passes?
18232 112724 001002 BNE 99$ ; not yet
18233 112726 000240 NOP ; debug aid
18234 112730 000452 BR TST26 ;;GO TO NEXT TEST
18235 112732 000240 99$: NOP
18236 112734 012737 000400 177746 MOV #400,CCR ;FLUSH CACHE
18237 112742 012701 112742 MOV #.,R1 ;START WITH CURRENT INSTRUCTION
18238 112746 005721 1$: TST (R1)+ ;READ A WORD
18239 112750 022701 113046 CMP #10$,R1 ;DONE?
18240 112754 001374 BNE 1$ ;IF NOT, CONTINUE
18241 112756 005737 020000 TST #20000 ;TO GUARANTY MISS ON 0
18242 112762 013702 000004 MOV #4,R2 ;STORE TIMEOUT VECTOR
18243 112766 012737 113022 000004 MOV #5$,#4 ;POINT NEW TO THE TEST
18244 112774 012737 000340 000006 MOV #340,#6 ;AT PRIORITY 7
18245 113002 042737 001000 177520 BIC #BIT09,BCSR ;DISABLE HALT ON BREAK
18246 113010 052737 000400 177520 BIS #BIT08,BCSR ;GO TO STANDALONE MODE
18247 113016 005737 000000 TST #0 ;MISS, SHOULD TIMEOUT
18248 113022 042737 000400 177520 5$: BIC #BIT08,BCSR ;CLEAR STANDALONE BIT
18249 113030 052737 001000 177520 BIS #BIT09,BCSR ;ENABLE HALT ON BREAK
18250 113036 022716 113022 CMP #5$, (SP) ;TIMEOUT?
18251 113042 001401 BEQ 10$ ;IF YES, BRANCH
18252 113044 104044 ERROR +44
18253 113046 012706 001100 10$: MOV #1100,SP ;RESTORE STACK
18254 113052 010237 000004 MOV R2,#4 ;AND TIMEOUT VECTOR
    
```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```

18256 .SBTTL TEST MOVING INVERSIONS TEST FOR DATA RAMS
18257 ;MOVING INVERSIONS TEST FOR DATA RAMS - THE TEST IS STARTED AFTER LOADING THE
18258 ;RAM STORE WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A
18259 ;1 IS SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE
18260 ;ADDRESS IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF
18261 ;THE WORD LEAVING THE ARRAY FILLED WITH 1'S. THE WHOLE PROCESS IS REPEATED
18262 ;PLUGGING IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESSING IN THE DOWNWARD
18263 ;DIRECTION. FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE
18264 ;LSB. TO SAVE TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING
18265 ;ONLY A SINGLE BIT AT A TIME EVERY FOURTH BIT WILL BE DONE CONCURRENTLY.
18266 ;THIS TEST RUNS IN STANDALONE MODE.
18267 ;
18268 ;BGNTST
18269 ;SETUP AND ENABLE MMU
18270 ;SETUP CCR TO ABORT PARITY ERRORS
18271 ;CLEAR CACHE
18272 ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
18273 ;LET FWDSEQ = #1
18274 ;LET ADDLSB = #1
18275 ;DO UNTIL ADDLSB EQ #20000
18276 ;. LET CURDAT = 0
18277 ;. LET RITEDA = #1
18278 ;. LET NEWDAT = #1
18279 ;. IF FWDSEQ = #1 THEN
18280 ;. . LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18281 ;. . LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18282 ;. ELSE
18283 ;. . LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18284 ;. . LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18285 ;. ENDIF
18286 ;. LET CURADD = FSTADD
18287 ;. LET DCOUNT = #0
18288 ;. SAVE CONTENTS OF VECTOR 114
18289 ;. LET VECTOR 114 POINT TO DATA STORE PARITY ABORT ROUTINE
18290 ;. DO UNTIL DCOUNT EQ #10
18291 ;. . LET RECDAT = @CURADD
18292 ;. . IF RECDAT NE CURDAT THEN
18293 ;. . . LET R1 EQUAL CURRENT DATA
18294 ;. . . GOTO DATA STORE PARITY ERROR ROUTINE
18295 ;. . ENDIF
18296 ;. . IF DCOUNT GT #3 THEN
18297 ;. . . LET @CURADD = @CURADD CLEARBY RITEDA
18298 ;. . ELSE
18299 ;. . . LET @CURADD = @CURADD SETBY RITEDA
18300 ;. . ENDIF
18301 ;. . LET RECDAT = @CURADD
18302 ;. . IF RECDAT NE NEWDAT THEN
18303 ;. . . LET R1 EQUAL NEW DATA
18304 ;. . . GOTO DATA STORE PARITY ERROR ROUTINE
18305 ;. . ENDIF
18306 ;. . IF CURADD EQ LASTAD THEN
18307 ;. . . IF RITEDA = #210 THEN
18308 ;. . . . IF DCOUNT NE #7 THEN
18309 ;. . . . . LET CURDAT = #377
18310 ;. . . . . LET RITEDA = #1
18311 ;. . . . . LET NEWDAT = #376
18312 ;. . . . ENDIF

```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```

18313      ;.      .      .      ELSE
18314      ;.      .      .      LET CURDAT = NEWDAT
18315      ;.      .      .      ROTATE RITEDA
18316      ;.      .      .      IF DCOUNT GT #3 THEN
18317      ;.      .      .      .      LET NEWDAT = NEWDAT CLEAREDBY RITEDA
18318      ;.      .      .      ELSE
18319      ;.      .      .      .      LET NEWDAT = NEWDAT SETBY RITEDA
18320      ;.      .      .      .      ENDIF
18321      ;.      .      .      ENDIF
18322      ;.      .      .      INCREMENT DCOUNT
18323      ;.      .      .      LET CURADD = FSTADD
18324      ;.      .      .      ELSE
18325      ;.      .      .      IF FWDSEQ = #1 THEN
18326      ;.      .      .      .      LET CURADD = CURADD + ADDLSB
18327      ;.      .      .      .      IF CARRY THEN
18328      ;.      .      .      .      .      LET CURADD = CURADD + #1
18329      ;.      .      .      .      .      ENDIF
18330      ;.      .      .      ELSE
18331      ;.      .      .      .      LET CURADD = CURADD - ADDLSB
18332      ;.      .      .      .      IF CARRY THEN
18333      ;.      .      .      .      .      LET CURADD = LASTAD - ADDLSB
18334      ;.      .      .      .      .      LET CURADD = CURADD - #1
18335      ;.      .      .      .      .      ENDIF
18336      ;.      .      .      .      ENDIF
18337      ;.      .      .      .      ENDIF
18338      ;.      .      .      ENDDO
18339      ;.      .      .      IF FWDSEQ EQ #1 THEN
18340      ;.      .      .      .      LET FWDSEQ = #0
18341      ;.      .      .      ELSE
18342      ;.      .      .      .      ROTATE ADDLSB
18343      ;.      .      .      .      LET FWDSEQ = #1
18344      ;.      .      .      .      ENDIF
18345      ;.      .      .      ENDDO
18346      ;.      .      .      ;RESTORE VECTOR 114
18347      ;.      .      .      ;ENDTST
18348
18349      ; *****
18350      113056 000004 TST26: SCOPE
18351      113060 000240      NOP
18352      113062 005737 003032      TST      CCHPAS      ;have done enough inclusive passes?
18353      113066 001003      BNE      99$      ; not yet
18354      113070 000240      NOP      ; debug aid
18355      113072 000137 114172      JMP      ENDMOV      ; yes skip this
18356      113076 000240      99$:      NOP
18357      113100 032777 000200 066032      BIT      #BIT07,#SWR      ;RUN THIS TEST?
18358      113106 001002      BNE      100$      ;IF SET, GO DO IT
18359      113110 000137 114172      JMP      ENDMOV      ;OTHERWISE, GO TO NEXT TEST
18360      113114 042737 001000 177520 100$:      BIC      #1000,BCSR      ;DISABLE HALT ON BREAK
18361      113122 004737 136574      JSR      PC,      INITMM      ;SETUP MEMORY MANAGEMENT
18362      113126 012737 002000 172354      MOV      #2000,KIPAR6      ;START ON 32K BOUNDARY
18363      113134 005237 177572      INC      SR0      ;TURN ON MMU
18364      113140 052737 000002 177746      BIS      #2,      CCR      ;SET DIAG. BIT
18365      113146 013737 000004 001160      MOV      @#4,      $TMPO      ;SAVE 4
18366
18367      ;STORE TEST IN THE FIRST 2K AND THEN IN THE SECOND 2K
18368

```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```

18369 113154 012703 140000      MOV      #140000,R3      ;START FOR THE TEST
18370 113160 012704 150000      MOV      #150000,R4      ;LOWER BOUNDARY TEST AREA
18371 113164 012705 157777      MOV      #157777,R5      ;HIGH BOUNDARY TEST AREA
18372 113170 000406              BR        2$              ;
18373 113172 012703 150000      1$: MOV      #150000,R3      ;START OF THE TEST
18374 113176 012704 140000      MOV      #140000,R4      ;LOWER BOUNDARY TEST AREA
18375 113202 012705 147777      MOV      #147777,R5      ;HIGH BOUNDARY
18376 113206 012702 113326      2$: MOV      #STMOVI,R2      ;START WITH CURRENT
18377 113212 010300      MCV      R3,R0           ;MOVE TO UPPER 4K
18378 113214 012220      3$: MOV      (R2)+,(R0)+     ;WORD BY WORD
18379 113216 022702 114172      CMP      #ENDMOV,R2      ;ALL DONE
18380 113222 001374              BNE      3$              ;
18381 113224 010400      MOV      R4,R0           ;CLEAR CACHE UNDER TEST
18382 113226 012701 004000      MOV      #4000, R1
18383 113232 005020      4$: CLR      (R1)+
18384 113234 077102      SOB      R1, 4$
18385 113236 004713      JSK      PC,(R3)         ;GO DO THE ROUTINE
18386 113240 005702      TST      R2              ;ANY ERRORS?
18387 113242 001411      BEQ      5$              ;IF NOT, CONTINUE
18388 113244 010037 114150      MOV      R0,RECDAT      ;DATA RECEIVED
18389 113250 010237 001122      MOV      R2,#BDADR      ;ADDRESS RECEIVED
18390 113254 042737 140000 001122 BIC      #140000,#BDADR  ;STRIP OF PAR BITS
18391 113262 104043      ERROR    +43            ;
18392 113264 000403      BR        6$              ;EXIT
18393 113266 022703 150000      5$: CMP      #150000,R3      ;DONE FOR BOTH HALVES?
18394 113272 001337      BNE      1$              ;IF NOT, DO AGAIN
18395 113274 052737 001000 177520 6$: BIS      #BIT09,BCSR    ;ENABLE HALT ON BREAK
18396 113302 013737 001160 000004 MOV      #TMP0,#4        ;RESTORE TIMEOUT VECTOR
18397 113310 012737 000400 177746 MOV      #400,CCR        ;INIT CCR FOR EXIT
18398 113316 005037 177572      CLR      SRO            ;TURN OFF MMU
18399 113322 000137 114172      JMP      ENDMOV          ;GO TO NEXT TEST
18400
18401      .DSABL AMA
18402 113326 052737 000400 177520 STMOVI: BIS      #BIT08,#BCSR    ;STANDALONE MODE
18403 113334 005002      CLR      R2              ;ERROR INDICATOR
18404 113336 012767 000001 000606 MOV      #1, FWDSEQ      ;INIT UPWARD ADDRESSING INDICATOR
18405 113344 012767 000001 000602 MOV      #1, ADDLSB      ;INIT LSB AND DO LOOP UNTIL SHIFTED OUT
18406 113352 042737 100000 172300 BIC      #100000,#KIPDRO ;NO BYPASS
18407 113360 012737 172360 000004 MOV      #KDPARO,#4      ;ALLOCATE TIMEOUT VECTOR
18408 113366 012737 000340 000006 MOV      #340, #6        ;AT PRIORITY 7
18409 113374 012737 000006 172360 MOV      #6, #KDPARO     ;PUT RETURN
18410 113402 052737 100000 172300 BIS      #100000,#KIPDRO ;BYPASS
18411 113410 005067 000546      TSTLUP: CLR      CURDAT   ;INIT CURRENT DATA
18412 113414 012767 000021 000534 MOV      #21, RITEDA     ;INIT DATA TO BE WRITTEN
18413 113422 012767 000021 000530 MOV      #21, NEWDAT     ;INIT EXPECTED DATA
18414 113430 005767 000516      TST      FWDSEQ         ;IF ADDRESSING UPWARD
18415 113434 001405      BEQ      1$              ;THEN
18416 113436 010467 000522      MOV      R4,FSTADD      ;LET FIRST ADDRESS EQUAL LOWEST VALUE
18417 113442 010567 000520      MOV      R5,LSTADD      ;LET LAST ADDRESS EQUAL HIGHEST VALUE
18418 113446 000404      BR        2$              ;ELSE
18419 113450 010567 000510      1$: MOV      R5,FSTADD      ;LET FIRST ADDRESS EQUAL HIGHEST VALUE
18420 113454 010467 000506      MOV      R4,LSTADD      ;LET LAST ADDRESS EQUAL LOWEST VALUE
18421 113460 016767 000500 000502 2$: MOV      FSTADD,CURADD  ;LET CURRENT ADDRESS EQUAL FIRST ADDRESS
18422 113466 005067 000452      CLR      DCOUNT         ;INIT LOOP COUNTER
18423 113472 022767 000010 000444 BGNTLP: CMP      #10, DCOUNT  ;DO LOOP 8 TIMES (4 TIMES TO WRITE 1'S
18424 113500 001567              BEQ      ENDTLP         ;AND 4 TIMES TO WRITE BACK 0'S
18425      ;

```


TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```

18483 114024 162767 007777 000136      SUB    #7777, CURADD      ;ROLL ADDRESS BACK
18484 114032 000411                    BR     10$               ;ELSE
18485 114034 166767 000114 000126 9$:  SUB    ADDLSB, CURADD     ;CALCULATE NEXT LOWER ADDRESS
18486 114042 020467 000122                    CMP    R4, CURADD        ;IF CURRENT ADDRESS HAS BEEN DECREASED
18487 114046 003403                    BLE    10$               ;BELOW LOWEST ADDRESS THEN
18488 114050 062767 007777 000112      ADD    #7777, CURADD     ;ROLL ADDRESS BACK
18489 114056 000605                    10$:  BR     BGNTLP        ;ENDDO
18490 114060 005767 000066      ENDTLP: TST  FWDSEQ       ;IF ADDRESSING UPWARD FINISHED
18491 114064 001404                    BEQ    1$                ;THEN
18492 114066 005067 000060      CLR    FWDSEQ           ;DO ADDRESSING DOWNWARD
18493 114072 000167 177312                    JMP    TSTLUP
18494 114076 012767 000001 000046 1$:  MOV    #1, FWDSEQ        ;SET ADDRESSING UPWARD INDICATOR
18495 114104 006'67 000044                    ROL    ADDLSB            ;UPDATE LSB TO NEXT POSITION
18496 114110 022767 020000 000036  ENDLUP: CMP    #20000, ADDLSB ;ALL DONE?
18497 114116 001406                    BEQ    EXITST           ;ENDDO
18498 114120 000167 177264                    JMP    TSTLUP
18499 114124 016700 000020      EXBAD: MOV    RECDAT, R0   ;STORE RECEIVED DATA
18500 114130 016702 000034                    MOV    CURADD, R2       ;STORE ADDRESS
18501 114134 042737 000400 177520  EXITST: BIC    #BIT08, @#BCSR ;OUT OF STANDALONE
18502 114142 000207                    RTS     PC
18503                                     .ENABL AMA
18504
18505 114144 000000      DCOUNT: .WORD 0
18506 114146 000000      EXPDAT: .WORD 0      ;STORES EXPECTED (GOOD) DATA FOR COMPARISONS
18507 114150 000000      RECDAT: .WORD 0      ;STORES RECIEVED DATA TO BE VERIFIED
18508 114152 000000      FWDSEQ: .WORD 0      ;USED TO INDICATE DIRECTION OF ADDRESSING
18509 114154 000000      ADDLSB: .WORD 0      ;STORES LEAST SIGNIFICANT BIT FOR RAM TESTS
18510 114156 000000      RITEDA: .WORD 0      ;STORES WRITE DATA FOR RAM TESTS
18511 114160 000000      NEWDAT: .WORD 0      ;DATA STORE FOR RAM TESTS
18512 114162 000000      CURDAT: .WORD 0      ;DATA STORE FOR RAM TESTS
18513 114164 000000      FSTADD: .WORD 0      ;STORES FIRST ADDRESS IN ADDRESSING SEQUENCE
18514 114166 000000      LSTADD: .WORD 0      ;STORES LAST ADDRESS IN ADDRESSING SEQUENCE
18515 114170 000000      CURADD: .WORD 0      ;STORES CURRENT ADDRESS FOR RAM TESTS
18516
18517 114172      ENDMOV:

```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

```

18519 .SBTTL TEST - MOVING INVERSIONS TEST FOR TAG STORE
18520 ;MOVING INVERSIONS TEST - THE TEST IS STARTED AFTER LOADING THE RAM STORE
18521 ;WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A 1 IS
18522 ;SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE ADDRESS
18523 ;IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF THE
18524 ;WORD LEAVING THE ARRAY FILLED WITH 1'S. THE WHOLE PROCESS IS REPEATED PLUGGING
18525 ;IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESS IN THE DOWARD DIRECTION.
18526 ;FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE LSB. TO SAVE
18527 ;TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING ONLY A SINGLE
18528 ;BIT AT A TIME EVERY THIRD BIT WILL BE DONE CONCURRENTLY. ALSO NOTE THAT
18529 ;SINCE THE TAG STORE CANNOT BE DIRECTLY ACCESSED THE ENTIRE PATTERN MUST BE
18530 ;DONE BY DOING MEMORY CYCLES TO THE CORRECT BUS ADDRESSES. TO DO THIS THE
18531 ;TEST WILL BE DONE WITH THE DIAGNOSTIC BIT IN THE CCR (BIT 1) SET TO A 1.
18532 ;THIS TEST RUNS IN STANDALONE MODE.
18533 ;
18534 ;
18535 ;BGNTST
18536 ;SETUP AND ENABLE MMU
18537 ;SETUP CCR TO ABORT PARITY ERRORS
18538 ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
18539 ;LET FWDSEQ = #1
18540 ;LET ADDLSB = #1
18541 ;DO UNTIL ADDLSB EQ #20000
18542 .. LET NEWDAT = 22200
18543 .. LET CURDAT = 0
18544 .. IF FWDSEQ = #1 THEN
18545 .. . LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18546 .. . LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18547 .. ELSE
18548 .. . LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18549 .. . LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18550 .. ENDIF
18551 .. LET DCOUNT = #0
18552 .. DO UNTIL DCOUNT EQ #10
18553 .. . READ CURADD USING CURDAT AS PAR
18554 .. . IF MISS THEN
18555 .. . . ERROR
18556 .. . ENDIF
18557 .. . WRITE CURADD USING NEWDAT AS PAR
18558 .. . ENDIF
18559 .. . IF HIT THEN
18560 .. . . ERROR
18561 .. . ENDIF
18562 .. . READ CURADD USING NEWDAT AS PAR
18563 .. . IF MISS THEN
18564 .. . . ERROR
18565 .. . ENDIF
18566 .. . IF CURADD EQ LASTAD THEN
18567 .. . . LET CURDAT = NEWDAT
18568 .. . . IF DCOUNT < #1 THEN
18569 .. . . . LET NEWDAT = NEWDAT SETBY RITEDA ROTATED LEFT
18570 .. . . ENDIF
18571 .. . . IF DCOUNT > #1 THEN
18572 .. . . . LET NEWDAT = NEWDAT CLR BY RITEDA ROTATED LEFT
18573 .. . . ELSE
18574 .. . . . LET NEWDAT = #155400 (NO ALL 1'S)
18575 .. . . . LET RITEDA = #22200

```


TEST - MOVING INVERSIONS TEST FOR TAG STORE

```

18632 114336 012220          3$:  MOV      (R2)+,(R0)+      ;WORD BY WORD
18633 114340 022702 115314    CMP      @ENDTAG,R2      ;ALL DONE?
18634 114344 001374          BNE      3$              ;
18635 114346 012700 114450    MOV      @STMOVT,R0      ;ADDRESS OF ROUTINE
18636 114352 162700 114134    SUB      @EXITST,R0      ;PROPER OFFSET
18637 114356 050003          BIS      R0,R3           ;
18638 114360 004713          JSR      PC,(R3)         ;GO DO THE ROUTINE
18639 114362 005700          TST      R0              ;ANY ERRORS?
18640 114364 001407          BEQ      4$              ;CONTINUE
18641 114366 010037 001122    MOV      R0,$BDADR       ;STORE FAILED ADDRESS
18642 114372 042737 160000 001122  BIC      @160000,@BDADR  ;CLEAR PAR
18643 114400 104050          ERROR   +50             ;
18644 114402 000403          BR       5$              ;EXIT TEST
18645 114404 022703 150000    4$:  CMP      @150000,R3     ;DONE FOR BOTH HALVES?
18646 114410 103736          BLO      1$              ;IF NOT, DO AGAIN
18647 114412 013737 001160 000004 5$:  MOV      $TMPO, @#4      ;RESTORE TIMEOUT VECTOR
18648 114420 012737 000400 177746  MOV      @400, CCR       ;INIT CCR FOR EXIT
18649 114426 005037 177572    CLR      SRO             ;TURN OFF MMU
18650 114432 005037 172516    CLR      MMR3           ;
18651 114436 052737 001000 177520  BIS      @1000,BCSR      ;ENABLE HALT ON BREAK
18652 114444 000137 115314    JMP      ENDTAG          ;EXIT TEST
18653
18654          .DSABL AMA
18655 114450 052737 000400 177520  STMOVT: BIS      @BIT08,@BCSR  ;STANDALONE MODE
18656 114456 042737 100000 172312  BIC      @BIT15,@KIPDR5  ;NO BYPASS
18657 114464 042737 100000 172300  BIC      @100000,@KIPDR0 ;NO BYPASS
18658 114472 012737 172360 000004  MOV      @KDPAR0,@#4     ;ALLOCATE TIMEOUT VECTOR
18659 114500 012737 000340 000006  MOV      @340, @#6       ;AT 7
18660 114506 012737 000006 172360  MOV      @6, @KDPAR0     ;PUT RETURN
18661 114514 052737 100000 172300  BIS      @100000,@KIPDR0 ;BYPASS
18662 114522 005037 172352    CLR      @KIPAR5        ;
18663 114526 010400          MOV      R4,R0          ;CLEAR CACHE UNDER TEST
18664 114530 012701 004000    MOV      @4000, R1      ;
18665 114534 005020          4$:  CLR      (R0)+         ;
18666 114536 077102          SOB      R1, 4$         ;
18667 114540 005000          CLR      R0             ;CLEAR ERROR FLAG
18668 114542 012767 000001 177402  MOV      @1, FWDSEQ      ;SET UPWARD ADDRESSING INDICATOR
18669 114550 012767 000002 177376  MOV      @2, ADDLSB     ;INITIALIZE LEAST SIGNIFIGANT BIT
18670 114556 005067 177400    TSLOOP: CLR      CURDAT  ;OLD DATA
18671 114562 012767 022200 177370  MOV      @22200,NEWDAT  ;SET ADDRESS BITS
18672 114570 012767 022200 177360  MOV      @22200,RITEDA  ;SET ADDRESS BITS
18673 114576 005767 177350    TST      FWDSEQ         ;IF ADDRESSING UPWARD
18674 114602 001405          BEQ      1$              ;THEN
18675 114604 010467 177354    MOV      R4, FSTADD     ;FIRST ADDRESS WILL BE LOWEST ADDRESS
18676 114610 010567 177352    MOV      R5, LSTADD     ;AND LAST ADDRESS WILL BE HIGHEST
18677 114614 000404          BR       2$              ;ELSE
18678 114616 010567 177342    1$:  MOV      R5, FSTADD     ;FIRST ADDRESS WILL BE HIGHEST ADDRESS
18679 114622 010467 177340    MOV      R4, LSTADD     ;AND LAST ADDRESS WILL BE LOWEST
18680 114626 005067 177312    2$:  CLR      DCOUNT        ;INITIALIZE LOOP COUNTER
18681 114632 016767 177326 177330  MOV      FSTADD, CURADD ;
18682
18683          ; DON'T REWRITE TIMEOUT VECTOR
18684
18685 114640 016700 177324    3$:  MOV      CURADD, R0     ;STORE CURRENT ADDRESS
18686 114644 042700 170000    BIC      @170000,R0     ;LEAVE ONLY LOW 4K BITS
18687 114650 022700 000004    CMP      @4, R0         ;TIMOUT VECTOR?
18688 114654 001526          BEQ      16$            ;IF SO, DON'T REWRITE IT

```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

18689	114656	022700	000006		CMP	#6, R0	;OR PRIORITY	
18690	114662	001523			BEQ	16#	;	
18691	114664	016737	177272	172352	MOV	CURDAT, @#KIPARS	;GET OLD PATTERN	
18692	114672	005777	177272		TST	@CURADD	;OLD DATA OK?	
18693	114676	013767	177752	177244	MOV	@#HITMIS,RECDAT		
18694	114704	032767	000004	177236	5#:	BIT	@BIT02, RECDAT	;IF ACCESS WAS A MISS
18695	114712	001002			BNE	6#	;THEN	
18696	114714	000167	000346		JMP	EXBAD2	;ERROR, EXIT	
18697	114720	016737	177234	172352	6#:	MOV	NEWDAT, @#KIPARS	;GET NEW PATTERN
18698	114726	005077	177236		CLR	@CURADD	;REGISTER AND WRITE LOCATION	
18699	114732	013767	177752	177210	MOV	@#HITMIS,RECDAT		
18700	114740	032767	000004	177202	8#:	BIT	@BIT02, RECDAT	;IF ACCESS WAS A HIT
18701	114746	001406			BEQ	9#	;THEN	
18702	114750	032767	000010	177172	10#:	BIT	@BIT03, RECDAT	;MISS?
18703	114756	001402			BEQ	9#	;IF SO, CONTINUE	
18704	114760	000167	000302		JMP	EXBAD2	;ERROR	
18705	114764	005777	177200		9#:	TST	@CURADD	;REGISTER AND READ LOCATION
18706	114770	013767	177752	177152	MOV	@#HITMIS,RECDAT		
18707	114776	032767	000004	177144	11#:	BIT	@BIT02, RECDAT	;IF ACCESS WAS A MISS
18708	115064	001002			BNE	12#	;THEN	
18709	115006	000167	000254		JMP	EXBAD2	;ERROR, EXIT	
18710	115012	026767	177150	177150	12#:	CMP	LSTADD, CURADD	;IF CURRENT ADDRESS IS LAST ADDRESS
18711	115020	001044			BNE	16#	;THEN	
18712	115022	016767	177132	177132	MOV	NEWDAT,CURDAT	;NEW KIPARS	
18713	115030	022767	000001	177106	CMP	#1, DCOUNT	;IF LOOP COUNTER LESS THAN 1	
18714	115036	003406			BLE	13#	;THEN	
18715	115040	006367	177112		ASL	RITEDA	;UPDATE OF NEW DATA....	
18716	115044	056767	177106	177106	BIS	RITEDA, NEWDAT	;BY SETTING BITS	
18717	115052	000421			BR	15#	;ENDIF	
18718	115054	022767	000001	177062	13#:	CMP	#1, DCOUNT	;CHANGE PATTERN?
18719	115062	001007			BNE	14#	;IF SO, BRANCH	
18720	115064	012767	022200	177064	MOV	@22200, RITEDA	;START WITH THE SAME	
18721	115072	012767	155400	177060	MOV	@155400,NEWDAT	;DON'T DO ALL 1'S	
18722	115100	000406			BR	15#	;	
18723	115102	006367	177050		14#:	ASL	RITEDA	;UPDATE OF NEW DATA....
18724	115106	046767	177044	177044	BIC	RITEDA, NEWDAT	;BY CLEARING BITS	
18725	115114	000400			BR	15#	;ELSE	
18726	115116	005267	177022		15#:	INC	DCOUNT	;INCREMENT THE LOOP COUNTER
18727	115122	016767	177036	177040	MOV	FSTADD, CURADD	;FINISH FIRST CURRENT ADDRESS	
18728	115130	000426			BR	18#	;ELSE	
18729	115132	005767	177014		16#:	TST	FWDSEQ	;IF ADDRESSING UPWARD
18730	115136	001412			BEQ	17#	;THEN	
18731	115140	066767	177010	177022	ADD	ADDLSB, CURADD	;ADD THE VALUE OF THE CURRENT LSB	
18732	115146	020567	177016		CMP	R5, CURADD	;IF CURRENT ADDRESS GREATER THAN HIGHEST	
18733	115152	002015			BGE	18#	;VIRTUAL ADDRESS THEN	
18734	115154	162767	007776	177006	SUB	@7776, CURADD	;ROLL IT BACK	
18735	115162	000411			BR	18#	;ELSE	
18736	115164	166767	176764	176776	17#:	SUB	ADDLSB, CURADD	;IF ADDRESSING DOWNWARD THEN SUBTRACT LSB
18737	115172	020467	176772		CMP	R4, CURADD	;IF CURRENT ADDRESS LESS THEN LOWEST	
18738	115176	003403			BLE	18#	;VIRTUAL ADDRESS THEN	
18739	115200	062767	007776	176762	ADD	@7776, CURADD	;ROLL IT BACK	
18740	115206	022767	000005	176730	18#:	CMP	#5, DCOUNT	;IF LOOP COUNTER LESS THAN 7
18741	115214	003402			BLE	181#	;THEN LOOP BACK FOR NEXT BIT POSITION	
18742	115216	000167	177416		JMP	3#		
18743	115222	005767	176724		181#:	TST	FWDSEQ	;IF ADDRESSING UPWARD
18744	115226	001404			BEQ	19#	;THEN	
18745	115230	005067	176716		CLR	FWDSEQ	;START ADDRESSING DOWNWARD	

TEST - PCR READ/WRITE BITS

```

18761 .SBTTL TEST PCR READ/WRITE BITS
18762 ;PCR AND BCSR READ/WRITE BITS
18763 ;THE FIRST TEST WILL CHECK THAT PCR REGISTER IS BOTH WORD AND
18764 ;BYTE ADDRESSABLE. BITS 14-09 AND 06-01 WILL BE WRITTEN AND READ
18765 ;AS ZEROES AND ONES. THE REST OF THE BITS HAVE TO BE ALL 0'S.
18766 ;ROUTINE TEST
18767 ;. SAVE PCR
18768 ;. LET PCR=0
18769 ;. DO FOR PATTERN=001111,110011,101010,010101
18770 ;. WRITE PCR<14-09>=PATTERN
18771 ;. WRITE PCR<06 01>=PATTERN
18772 ;. IF PCR<14 09> NE PATTERN OR PCR<06 01> NE PREVIOUS
18773 ;. PATTERN
18774 ;. THEN ERROR
18775 ;. ENDF
18776 ;. WRITE PCR<06-01>=PATTERN
18777 ;. IF PCR<14-09> NE PATTERN OR PCR<06-01> NE PATTERN
18778 ;. THEN ERROR
18779 ;. ENDF
18780 ;. ENDDO
18781 ;. WRITE PCR=0101010101010101
18782 ;. IF PCR NE 0101010001010100 THEN
18783 ;. ERROR
18784 ;. ENDF
18785 ;ENDROUTINE
18786
18787 ;:*****

```

```

18788 115314 000004
18788 115316 005037 177522
18789 115322 012702 115461
H BYTE
18790 115326 012704 115460
18791 115332 012703 000004
18792
18793
18794
18795 115336 111237 177523
18796 115342 121237 177523
18797 115346 001003
18798 115350 121437 177522
18799 115354 001405
18800 115356 111237 001125
18801 115362 111437 001124
18802 115366 104051
18803 115370 105724
18804
18805
18806
18807 115372 111237 177522
18808 115376 121237 177522
18809 115402 001003
18810 115404 121237 177522
18811 115410 001405
18812 115412 111237 001124
18813 115416 111237 001124
18814 115422 104051
18815 115424 105722
18816 115426 077335

TST30: SCOPE
        CLR     PCR                ;INITIALIZE PCR TO 0
        MOV     #SIXBIT+1,R2       ;R2->TABLE OF PATTERNS FOR 6 R/W BITS IN EAC
        MOV     #SIXBIT,R4
        MOV     #4,R3              ;R3 POINTER TO PREVIOUS PATTERN
        ;DO 4 TIMES
; WRITE TO HIGH BYTE FIRST
1$:     MOVB    (R2),PCR+1          ;WRITE TO HIGH BYTE
        CMPB   (R2),PCR+1         ;BYTE WRITTEN OK?
        BNE    2$                 ;IF NOT, BRANCH
        CMPB   (R4),PCR           ;LOW BYTE CHANGED?
        BEQ    3$                 ;IF NOT, BRANCH
2$:     MOVB    (R2),$GDDAT+1      ;EXPECTED PATTERN HIGH BYTE
        MOVB   (R4),$GDDAT        ;LOW BYTE
        ERROR  +51                ;ERROR PCR READ/WRITE BITS
3$:     TSTB   (R4)+              ;INCREMENT POINTER FOR OLD
; WRITE TO LOW BYTE
;
        MOVB   (R2),PCR           ;WRITE TO LOW BYTE
        CMPB   (R2),PCR           ;BYTE WRITTEN OK?
        BNE    4$                 ;IF NOT, BRANCH
        CMPB   (R2),PCR           ;HIGH BYTE CHANGED?
        BEQ    5$                 ;IF NOT, BRANCH
4$:     MOVB    (R2),$GDDAT        ;EXPECTED PATTERN HIGH BYTE
        MOVB   (R2),$GDDAT        ;LOW BYTE
        ERROR  +51                ;ERROR PCR READ/WRITE BITS
5$:     TSTB   (R2)+              ;INCREMENT POINTER FOR NEW PATTERN
        SOB    R3,1$             ;DO FOR ALL 4 PATTERNS

```

TEST - PCR READ/WRITE B' J

```

18817
18818
18819
18820 115430 012737 052525 177522      MOV    #52525,PCR      ;WRITE A PATERN
18821 115436 022737 052124 177522      CMP    #52124,PCR     ;ALL BUT BITS <8,7,0> OK?
18822 115444 001404                      BEQ    6$             ;IF SO, BRANCH
18823 115446 112737 052124 001124      MOVB  #52124,$GDDAT  ;EXPECTED PATTERN
18824 115454 104051                      ERROR  +51           ;ERROR PCR READ/WRITE BITS
18825 115456                      6$: BR    TST31      ;;GO TO NEXT TEST
18826
18827 115460 000 036 146 SIXBIT: .BYTE 0,36,146,124,52 ;001111,110011,101010,010101
18828 115463 124 052
18829
18830 .EVEN ;BINARY DIVIDE FOR 6 BITS

```

TEST - BCSR READ/WRITE BITS

```

18832 .SBTTL TEST - BCSR READ/WRITE BITS
18833 ;THE SECOND TEST WILL CHECK THAT BCSR<7-5;2-0> CAN BE WRITTEN AND
18834 ;READ AS ZEROES AND ONES. BCSR<14,03> SHOULD BE 0'S. BCSR<04>
18835 ;SHOULD BE CLEARED BY RESET INSTRUCTION.
18836 ;ROUTINE TEST
18837 ;. FOR PATTERN=011,010,101 DO
18838 ;. WRITE BCSR<7-5>=PATTERN
18839 ;. IF BCSR<7-5> NE PATTERN THEN
18840 ;. ERROR
18841 ;. ENDF
18842 ;. WRITE BCSR<2-0>=PATTERN
18843 ;. IF BCSR<2-0> NE PATTERN THEN
18844 ;. ERROR
18845 ;. ENDF
18846 ;. ENDDO
18847 ;. IF BCSR<14,03> NE <0,0> THEN
18848 ;. ERROR
18849 ;. ENDF
18850 ;. LET BCSR<04>=1
18851 ;. IF BCSR<04> NE #1 THEN
18852 ;. ERROR
18853 ;. ENDF
18854 ;. EXECUTE "RESET"
18855 ;. IF BCSR<04> NE 0 THEN
18856 ;. ERROR
18857 ;. ENDF
18858 ;. LET BCSR<04>=0 (THIS BIT IS WRITE ENABLE FOR EAROM)
18859 ;ENDROUTINE
18860
18861 ;*****
18862 115466 000004 TST31: SCOPE
18863 115470 013737 177520 002730 MOV BCSR,SAVBR ;SAVE BCSR
18864 115476 005037 177520 CLR BCSR ;CLEAR BCSR
18865 ;
18866 ; WRITE TO BITS <7-5> AND <2-0>
18867 ;
18867 115502 012703 115726 MOV #THRBIT,R3 ;POINTER FOR PATTERN TABLE
18868 115506 005037 001124 CLR #GDDAT ;CLEAR A LOCATION
18869 115512 012702 000003 MOV #3,R2 ;DO FOR ALL PATTERNS
18870 115516 111337 177520 1$: MOVB (R3),BCSR ;WRITE TO BITS <7-5>
18871 115522 111337 001124 MOV# (R3),#GDDAT ;EXPECTED PATTERN
18872 115526 122337 177520 CMPB (R3)+,BCSR ;BITS WRITTEN OK?
18873 115532 001401 BEQ 2$ ;IF SO, BRANCH
18874 115534 104052 ERROR +52 ;ERROR IN BCSR READ/WRITE BITS
18875 115536 111337 177520 2$: MOVB (R3),BCSR ;WRITE TO BITS <2-0>
18876 115542 111337 001124 MOVB (R3),#GDDAT ;EXPECTED PATTERN FOR ERRORS
18877 115546 122337 177520 CMPB (R3)+,BCSR ;BITS WRITTEN OK?
18878 115552 001401 BEQ 3$ ;IF SO, BRANCH
18879 115554 104052 ERROR +52 ;ERROR IN BCSR READ/WRITE BITS
18880 115556 077221 3$: SOB R2,1$ ;CONTINUE TILL ALL PATTERNS DONE
18881 ;
18882 ; CHECK UNUSED BITS <3>
18883 ;
18884 115560 012737 000010 177520 MOV #10,BCSR ;WRITE TO BIT <3>
18885 115566 032737 000010 177520 BIT #BIT03,BCSR ;ALL ZEROES?
18886 115574 001403 BEQ 4$ ;IF YES, BRANCH
18887 115576 005037 001124 CLR #GDDAT ;EXPECTED PATTERN

```

TEST - BCSR READ/WRITE BITS

```

18888 115602 104052          ERROR +52          ;ERROR IN BCSR READ/WRITE BITS
18889                               ;
18890                               ; CHECK THAT BIT <4> CLEARS BY RESET
18891                               ;
18892 115604 052737 000020 177520 4$:  BIS    #BIT04,BCSR          ;SET BIT 4
18893 115612 032737 000020 177520      BIT    #BIT04,BCSR          ;WRITTEN OK?
18894 115620 001005                BNE    5$                    ;IF SO BRANCH
18895 115622 012737 000020 001124      MOV    #BIT04,$GDDAT        ;EXPECTED PATTERN
18896 115630 104052          ERROR +52          ;ERROR IN BCSR READ/WRITE BITS
18897 115632 000415                BR     7$                    ;EXIT TEST
18898 115634 042737 000020 177520 5$:  BIC    #BIT04,BCSR          ;TRY TO CLEAR BIT 4
18899 115642 032737 000020 177520      BIT    #BIT04,BCSR          ;CLEARED OK?
18900 115650 001403                BEQ    6$                    ;IF SO BRANCH
18901 115652 005037 001124          CLR    $GDDAT              ;EXPECTED PATTERN
18902 115656 104052          ERROR +52          ;ERROR IN BCSR READ/WRITE BITS
18903 115660 005737 001206          6$:  TST    $PASS                ;FIRST PASS?
18904 115664 001014                BNE    8$                    ;IF NOT FIRST PASS,EXIT
18905 115666 052737 000020 177520 7$:  BIS    #BIT04,BCSR          ;SET BIT 4 AGAIN
18906 115674 000005                RESET                   ;EXECUTE RESET
18907 115676 032737 000020 177520      BIT    #BIT04,BCSR          ;BIT 4 CLEARED?
18908 115704 001404                BEQ    8$                    ;IF YES, BRANCH
18909 115706 104053          ERROR +53          ;RESET DOESN'T CLEAR BCSR<4>
18910 115710 042737 000020 177520      BIC    #BIT04,BCSR          ;CLEAR BIT 4
18911 115716 013737 002730 177520 8$:  MOV    SAVBR,BCSR          ;RESTORE BCSR
18912 115724 000403                BR     TST32                ;;GO TO NEXT TEST
18913
18914 115726 140 003 100 THRBIT: .BYTE 140,3,100,2,240,5 ;011,010,101 FOR BITS <7-5,2 0>
      115731 002 240 005
18915

```


TEST - 16 BIT ROM CHECKSUM TEST

```

18917 .SBTTL TEST 16 BIT ROM CHECKSUM TEST
18918 ;ROM'S CHECKSUMS
18919 ;
18920 ;16 BIT ROM TEST
18921 ;THE FIRST TEST WILL CLEAR BCSR<07>, LOAD PCR<14-09> WITH
18922 ;ROM ADDRESS BITS <14-09>, AND CHECK CHECKSUMS OF 16-BIT ROM
18923 ;BY ACCESSING IT THRU BUS ADDRESSES 173000-173776. THEN WITH
18924 ;BCSR<06;05> BOTH CLEAR, PCR<06-01> USED AS ADDRESS BITS 14-09,
18925 ;THE SAME THING WILL BE DONE BY ADDRESSING 16-BIT ROM THRU BUS ADDRESSES
18926 ;165000-165776. THE RESULTS SHOULD BE THE SAME AND SHOULD COMPARE
18927 ;WITH THAT STORED IN THE BOOT AND DIAGNOSTIC ROM.
18928 ;
18929 ;BCSR <07> DISABLE 17773000
18930 ; <06> DISABLE 17765000
18931 ; <05> ROM SOCKET 3 AT 17765000
18932 ;
18933 ;ROUTINE TEST
18934 ;. LET BCSR<7,6,5>=0,0,0
18935 ;. DO FOR R1 FROM #0 TO #31. BY #1 DO
18936 ;. . LET PCR<14-09>=R1
18937 ;. . DO FOR R2 FROM #0 TO #776 BY 2
18938 ;. . . CALCULATE CHECKSUM THRU 173000
18939 ;. . ENDDO
18940 ;. ENDDO
18941 ;. IF CHECKSUM NE #0 THEN
18942 ;. . ERROR
18943 ;. ENDIF
18944 ;. DO FOR R1 FROM #0 TO #31. BY #1
18945 ;. . LET PCR<06-01>=R1
18946 ;. . DO FOR R2 FROM #0 TO #776 BY 2
18947 ;. . . CALCULATE CHECKSUM THRU 165000
18948 ;. . ENDDO
18949 ;. ENDDO
18950 ;. IF CHECKSUM NE #0 THEN
18951 ;. . ERROR
18952 ;. ENDIF
18953 ;ENDROUTINE
18954 ;*****
18955 ;*****
18956 115734 000004 TST32: SCOPE
18957 115736 013737 177520 002730 MOV BCSR,SAVBR ;SAVE BCSR
18958 115744 042737 000340 177520 BIC #BIT07!BIT06!BIT05,BCSR ;READ 16 BIT ROM
18959 115752 052737 001000 177520 BIS #1000,BCSR ; enable H08, for AP1
18960 ;
18961 ; CALCULATE LOW BYTE CHECKSUM'S THRU 173000 AND 165000
18962 ;
18963 115760 005001 CLR R1 ;PAGE COUNT FOR ALL BK
18964 115762 005037 177522 CLR PCR ;CLEAR PAGE CONTROL REGISTER
18965 115766 005037 002724 1# CLR ACTCHS ;CLEAR CHECKSUM AT 173000
18966 115772 005037 001160 CLR $TMPO ;AT 165000
18967 115776 005002 CLR R2 ;CLEAR COUNTER THRU A PAGE
18968 116000 016203 173000 2# MOV 173000(R2),R3 ;GET LOW BYTE THRU 173000
18969 116004 016204 165000 MOV 165000(R2),R4 ;THRU 165000
18970 116010 060337 002724 ADD R3,ACTCHS ;CALCULATE CHECKSUM THRU 173000
18971 116014 060437 001160 ADD R4,$TMPO ;CALCULATE CHECKSUM THRU 165000
18972 116020 005722 TST (R2)+ ;GET NEXT WORD
18973 116022 022702 000776 CMP #776,R2 ;WORD BEFORE LAST?

```


TEST 8 BIT EEROM CHECKSUM (105dec bytes) TEST

```

19026 .SBTTL TEST - 8 BIT EEROM CHECKSUM (105dec bytes) TEST
19027 ;THIS TEST WILL CLEAR BCSR<6>, SET BCSR<5>, LOAD PCR<5-1>
19028 ;WITH ADDRESS BITS 13-09, AND CHECK CRC PATTERNS OF 8-BIT EEROM BY
19029 ;ACCESSING IT THRU ADDRESSES 165000-165776. THIS TEST VERIFIES THE
19030 ;RESPONSE ONLY OF THE BASE AREA.
19031 ;ROUTINE TEST
19032 ;. LET BCSR<5>=1
19033 ;. LET OLDCRC=0
19034 ;. LET PCR<05-01>=0
19035 ;. CALCULATE CHECKSUM FOR THE FIRST 320 LOCATIONS
19036 ;. IF RESULTING CHECKSUM NOT ZERO THEN
19037 ;. ERROR
19038 ;. ENDIF
19039 ;ENDROUTINE
19040
19041 ;*****
19042 116254 000004 TST33: SCOPE
19043 116256 013737 177520 002730 MOV BCSR,SAVBR ;SAVE BCSR
19044 116264 042737 000100 177520 BIC #BIT06,BCSR ;ENABLE INTERNAL RESPONSE
19045 116272 052737 000040 177520 BIS #BIT05,BCSR ;SELECT 8-BIT ROM
19046 116300 005037 001160 CLR #TMP0 ;CLEAR SUM
19047 ;
19048 ; CALCULATE LOW BYTE CHECKSUM'S THRU 165000
19049 ;
19049 116304 005001 1#: CLR R1 ;PAGE COUNT FOR ALL 8K
19050 116306 005037 177522 CLR PCR ;CLEAR PAGE CONTROL REGISTER
19051 116312 005037 002724 2#: CLR ACTCHS ;CLEAR CHECKSUM AT 165000
19052 116316 005002 CLR R2 ;CLEAR COUNTER THRU A PAGE
19053 116320 116204 165000 3#: MOV# 165000(R2),R4 ;GET A BYTE THRU 165000
19054 116324 060437 001160 ADD R4,#TMP0 ;CALCULATE CHECKSUM THRU 165000
19055 116330 005722 TST (R2)+ ;GET NEXT WORD
19056 116332 022702 000316 CMP #316,R2 ;WORD BEFORE LAST?
19057 116336 001004 BNE 4# ;IF NOT, BRANCH
19058 116340 122704 000252 CMP# #252,R4 ;314 SHOULD HAVE 252
19059 116344 001765 BEQ 3# ;IF YES, BRANCH
19060 116346 104055 ERROR +55 ;PAGE NUMBER STORED WRONG
19061 116350 022702 000322 4#: CMP #322,R2 ;LAST WORD IN A PAGE?
19062 116354 003361 BGT 3# ;IF NOT, BRANCH
19063 116356 113737 165010 001162 MOV# 165010,#TMP1 ;STORE SIZE.<3>=1 2K
19064 116364 105737 001160 TST #TMP0 ;CHECKSUM 0 AT 165000?
19065 116370 001401 BEQ 5# ;IF YES, BRANCH
19066 116372 104055 ERROR +55 ;IN CHECKSUM AT 165000
19067 116374 005037 177522 5#: CLR PCR
19068 116400 013737 002730 177520 MOV SAVBR,BCSR ;RESTORE BCSR
19069

```


TEST - 8-BIT EEROM READ-WRITE - TEST

```

19127 116424 000240          NOP
19128          ;          CMPB    #APTENV,#ENV          ; no, do again
19129          ;          BEQ     1$              ; IN APT MODE?
19130 116426 032777 010000 062504 ;          BIT     #BIT12,#SWR          ; <IF YES, EXIT TEST>
19131 116434 001016          BNE     2$              ; bit 12 (do EEROM test) set?
19132 116436 000240          NOP
19133 116440 000137 116770 1$:      JMP     TSTEND          ; exit point for program
19134 116444 015 012 116 33$:    .ASCIZ  <15><12>/NOW TESTING EEROM/<15><12>
      116447 117 127 040
      116452 124 105 123
      116455 124 111 116
      116460 107 040 105
      116463 105 122 117
      116466 115 015 012
      116471 000

19135          .EVEN
19136          ; save base area, switch registers
19137
19138 116472 104401 116444 2$:    TYPE    ,33$          ; INDICATE EEROM TEST UNDERWAY
19139 116476 017737 062436 003050 MOV    #SWR,SAVSWR      ; save softswitch settings
19140 116504 042777 041777 062426 BIC    #41777,#SWR      ; clear any interfering
19141 116512 013737 177520 002730 MOV    BCSR,SAVBR       ;SAVE REGISTER
19142 116520 052737 001060 177520 BIS    #1060,BCSR       ;ENABLE INTERNAL ROM'S
19143 116526 000240          NOP          ;ENABLE 8-BIT ROM, HOB
19144 116530 012701 143762 MOV    #POWER+10,R1     ;LAST LOCATION IN PROGRAM
19145 116534 005037 177522 CLR    PCR              ;START WITH PAGE 0
19146 116540 005002          CLR    R2              ;DISPLACEMENT 0
19147 116542 016221 165000 3$:    MOV    165000(R2),(R1)+ ;STORE A WORD
19148 116546 005722          TST    (R2)+           ;GET NEXT WORD
19149 116550 022702 000334 CMP    #334,R2          ;ALL 332 LOCATIONS DONE?
19150 116554 003372          BGT    3$              ;IF NOT, CONTINUE
19151
19152          ; test 2K section
19153
19154 116556 112703 000252 MOVB   #252,R3           ; first pattern, 10 101 010
19155 116562 005037 003022 CLR    ERRCNT           ; zero the cumulative error count
19156
19157 116566 000240          201$:  NOP
19158 116570 005037 177522 CLR    PCR              ; clears page register (start @ 165000)
19159 116574 000240          202$:  NOP
19160 116576 005002 CLR    R2              ; first location each page of EEROM
19161 116600 000240          203$:  NOP
19162 116602 110362 165000 MOVB   R3,165000(R2)     ; write the test pattern
19163 116606 004737 116754 JSR    PC,DELAY         ; wait for write time
19164 116612 120362 165000 CMPB   R3,165000(R2)     ; read back the written word
19165 116616 001413 BEQ    204$            ; if O. K. readback, skip handle error
19166
19167 116620 005237 003022 INC    ERRCNT           ; update cumulative error count
19168 116624 116237 165000 001126 MOVB   165000(R2),#BDDAT ; put the read data in display area
19169 116632 013704 177522 MOV    PCR,R4
19170 116636 010237 001122 MOV    R2,#BDADR       ; also address location info
19171 116642 104134 ERROR  +134            ; do EEROM read/write error report
19172 116644 000240          NOP          ; continue testing
19173
19174 116646 005722          204$:  TST    (R2)+           ; last location checked in a page
19175 116650 022702 000776 CMP    #776,R2          ;
19176 116654 003351          BGT    203$            ; continue till all page loc. tested

```

TEST - 8-BIT EAROM READ-WRITE - TEST

```

19177 116656 062737 000002 177522      ADD    #2,PCR          ; change PCR every 512 dec. bytes
19178 116664 122737 000020 177522      CMPB   #20,PCR         ; 8 (256 Byte pgs in 2K) * 2 (holes)=20
19179 116672 001340                BNE    205#           ; finish page
19180 116674 122703 000125                CMPB   #125,R3        ; test for both patterns written
19181 116700 001403                BEQ    205#           ; Exit point
19182 116702 112703 000125                MOVB   #125,R3        ; invert the test pattern
19183 116706 000727                BR     201#           ; do the test over with new pattern
19184
19185
19186 116710 000240                205#:  NOP            ; test over
19187
19188 116712 012701 143762                MOV    #POWER+10,R1   ;LAST LOCATION IN PROGRAM
19189 116716 005037 177522                CLR    PCR            ;START WITH PAGE 0
19190 116722 005002                CLR    R2             ;DISPLACEMENT 0
19191 116724 012162 165000                11#:  MOV    (R1)+,165000(R2) ;RESTORE A WORD
19192 116730 004737 116754                JSR    PC, DELAY      ; wait for write time
19193 116734 005722                TST    (R2)+          ;GET NEXT WORD
19194 116736 022702 000334                CMP    #334,R2        ;ALL 332 LOCATIONS DONE?
19195 116742 003370                BGT    11#            ;IF NOT, CONTINUE
19196 116744 013737 002730 177520                MOV    SAVBR,BCSR     ;RESTORE REGISTER
19197 116752 000406                BR     TSTEND         ; goto next test (swr restored there)
19198
19199      ; ----- SUBROUTINES AREA -----
19200      ; the following subroutine causes a delay of a certain no of milliseconds
19201      ; the number of ms. delayed is based on the type of EEROM as indicated by R5
19202
19203 116754 010046                DELAY: MOV    R0,-(SP)   ; save R0
19204 116756 012700 023420                MOV    #10000., R0    ; ALL ELSE = 10 MS DELAY
19205 116762 077001                1#:  SOB    R0, 1#     ; actual delay
19206 116764 012600                MOV    (SP)+, R0      ; restore R0
19207 116766 000207                RTS    PC
19208
19209
19210 116770 000240                TSTEND: NOP           ; everything done
19211

```


TEST - LKS BIT 7

19269	117114	012702	000003		MOV	03,R2				
19270	117120	012701	077777	4:	MOV	077777,R1				;DO 3 TIMES TO SYNCHRONISE
19271	117124	105737	177546	5:	TSTB	LKS				;COUNTER FOR SLOW CLOCKS
19272	117130	100401			BMI	6:				;READY LKS<7>=1?
19273	117132	077104			SOB	R1,5:				;IF SO, DO NEXT LOOP
19274	117134	105737	177546	6:	TSTB	LKS				;OTHERWISE, GO THRU COUNT
19275	117140	100401			BMI	7:				;WAS READY 1?
19276	117142	104057			ERROR	+57				;IF YES, BRANCH
19277	117144	077213		7:	SOB	R2,4:				;LKS<07> DOES NOT BECOME 1
19278	117146	005737	002722	8:	TST	LKSFL				;DO ALL 3 TIMES
19279	117152	001401			BEQ	TST36				;ANY INTERRUPTS W/O LKS<6>=1?
19280	117154	104061			ERROR	+61				TEST
19281										;:IF NONE, EXIT
										;ILLEGAL CLOCK INTERRUPTS

TEST LKS INTERRUPT PRIORITY

```

19283 .SBTTL TEST - LKS INTERRUPT PRIORITY
19284 ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
19285 ;AND DON'T HAPPEN AT PRIORITY 6.
19286 ;ROUTINE TEST
19287 ;IF UFD AND LKS IS DISABLED THEN
19288 ;. EXIT TEST
19289 ;ENDIF
19290 ;. SET PRIORITY TO 5
19291 ;. CLEAR INTERRUPT_FLAG
19292 ;. LET LKS<06>=#1 (ENABLE INTERRUPTS)
19293 ;. SET COUNTER TO WAIT FOR 3 INTERRUPTS
19294 ;. REPEAT
19295 ;. . DECREMENT COUNTER
19296 ;. UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
19297 ;. CLEAR LKS<06>
19298 ;. IF LKS<07> EQ #1 THEN
19299 ;. . ERROR (WAS NOT CLEARED ON INTERRUPT)
19300 ;. ENDF
19301 ;. IF COUNTER LT TIME_REQUIRED_FOR_3 INTERRUPTS_FOR 800HZ
19302 ;. . ERROR (INTERRUPTS NEVER GO LOW)
19303 ;. ENDF
19304 ;. IF INTERRUPT_FLAG LT #3 THEN
19305 ;. . ERROR (INTERRUPTS DON'T HAPPEN)
19306 ;. ENDF
19307 ;. CLEAR INTERRUPT_FLAG
19308 ;. WAIT FOR LKS<7>=1
19309 ;. LET LKS<7>=0
19310 ;. IF LKS<7> NE #0 THEN
19311 ;. . ERROR (LKS<7> NOT CLEARED)
19312 ;. ENDF
19313 ;. SET PRIORITY TO 6
19314 ;. SET COUNTER TO 1 SLOW CLOCK INTERRUPT
19315 ;. SET LKS<06>
19316 ;. REPEAT
19317 ;. . DECREMENT COUNTER
19318 ;. UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
19319 ;. IF INTERRUPT_FLAG NE #0 THEN
19320 ;. . ERROR (INTERRUPT WAS AT WRONG PRIORITY)
19321 ;. ENDF
19322 ;. RESTORE ORIGINAL PRIORITY
19323 ;ENDROUTINE
19324 ;
19325 ;ROUTINE LINE_CLOCK_INTERRUPT
19326 ;. INCREMENT INTERRUPT_FL
19327 ;ENDROUTINE
19328 ;
19329 ;*****
19330 117156 000004 TST36: SCOPE
19331 117160 032737 000100 000052 BIT #BIT06,#52 ;UFD MODE?
19332 117166 001404 BEQ 1# ;IF NOT, GO DO TEST
19333 117170 032737 010000 177520 BIT #BIT12,BCSR ;LKS DISABLED?
19334 117176 001132 BNE TST37 ;;IF DISABLED, EXIT TEST
19335 ;
19336 ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
19337 117200 042737 000100 177546 1#: BIC #BIT06,LKS ; FROM END OF TEST 42?? PROBLEM
19338 117206 005037 002722 CLR LKSFL ;CLEAR INTERRUPT FLAG
    
```

TEST - LKS INTERRUPT PRIORITY

```

19339 117212 012737 137062 000100      MOV      #LKSINT,100      ;POINT VECTOR TO ROUTINE
19340 117220 012701 077777      MOV      #77777,R1      ;COUNTER FOR SLOW CLOCK
19341 117224 052737 000100 177546      BIS      #BIT06,LKS      ;SET INTERRUPT ENABLE BIT
19342 117232 032737 000100 177546      BIT      #BIT06,LKS      ;BIT SET OK?
19343 117240 001001      BNE      2$              ;IF YES, BRANCH
19344 117242 104131      ERROR    +131           ;ERROR WRITING 1 TO LKS<6>
19345 117244 106427 000240      2$:      MTPS     #240           ;SET PRIORITY TO 5
19346 117250 022737 000003 002722 3$:      CMP      #3,LKSFL      ;3 INTERRUPTS HAPPENED?
19347 117256 001401      BEQ      4$              ;IF YES, BRANCH
19348 117260 077105      SOB      R1,3$          ;STAY IN A LOOP
19349
19350      ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
19351
19352 117262 042737 001000 177520 4$:      BIC      #1000,BCSR     ;DISABLE HALT ON BREAK
19353 117270 106427 000340      MTPS     #340           ;RAISE PRIORITY
19354 117274 042737 000100 177546      BIC      #BIT06,LKS     ;DISABLE INTERRUPTS
19355 117302 032737 000200 177546      BIT      #BIT07,LKS     ;LKS<7> CLEARED AFTER INTERRUPTS?
19356 117310 001401      BEQ      5$              ;IF 0, BRANCH
19357 117312 104062      ERROR    +62           ;INTERRUPTS DON'T CLEAR LKS<7>
19358 117314 105737 177546      5$:      TSTB     LKS            ;LKS<7>=1?
19359 117320 100375      BPL      5$              ;IF NOT, WAIT
19360 117322 005037 177546      CLR      LKS            ;CLEAR LKS<7>
19361 117326 032737 000200 177546      BIT      #BIT07,LKS     ;LKS<7> CLEARED?
19362 117334 001401      BEQ      6$              ;IF YES, BRANCH
19363 117336 104060      ERROR    +60           ;LKS<7> NOT CLEARED ON WRITE
19364 117340 032737 000100 177546 6$:      BIT      #BIT06,LKS     ;LKS<6>=0?
19365 117346 001401      BEQ      7$              ;IF YES, BRANCH
19366 117350 104131      ERROR    +131           ;ERROR WRITING 0 TO LKS<6>
19367 117352 052737 001000 177520 7$:      BIS      #1000,BCSR     ;ENABLE HALT ON BREAK
19368 117360 022701 077737      CMP      #77737,R1      ;COUNTER AT LESS THAN 800HZ?
19369 117364 002001      BGE      8$              ;IF NOT, BRANCH
19370 117366 104063      ERROR    +63           ;READY LINE DOES NOT GO LOW
19371 117370 022737 000003 002722 8$:      CMP      #3,LKSFL      ;DID 3 INTERRUPTS HAPPEN?
19372 117376 001404      BEQ      9$              ;IF YES, BRANCH
19373 117400 012737 000003 001124      MOV      #3,#GDDAT      ;3 INTERRUPTS EXPECTED
19374 117406 104064      ERROR    +64           ;INTERRUPTS DON'T HAPPEN
19375
19376      ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
19377
19378 117410 005037 002722 9$:      CLR      LKSFL          ;CLEAR INTERRUPT FLAG
19379 117414 106427 000300      MTPS     #300           ;RAISE PRIORITY TO 6
19380 117420 012701 077777      MOV      #77777,R1      ;COUNTER FOR SLOW CLOCK
19381 117424 052737 000100 177546      BIS      #BIT06,LKS     ;SET INTERRUPT ENABLE BIT
19382 117432 005737 002722 10$:     TST      LKSFL          ;ANY INTERRUPTS?
19383 117436 001001      BNE      11$            ;IF YES, EXIT LOOP
19384 117440 077104      SOB      R1,10$         ;CONTINUE WITH COUNT
19385 117442 005737 002722 11$:     TST      LKSFL          ;ANY INTERRUPTS?
19386 117446 001404      BEQ      12$            ;IF NO, BRANCH
19387 117450 012737 000005 001124      MOV      #5,#GDDAT      ;STORE PRIORITY FOR TYPE OUT
19388 117456 104065      ERROR    +65           ;INTERUPTS HAPPEN AT WRONG PRIORITY
19389 117460 106427 000340      12$:     MTPS     #340           ;RESTORE PRIORITY
19390

```

TEST - LINE CLOCK DISABLE

```

19392 .SBTTL TEST - LINE CLOCK DISABLE
19393
19394 ;LINE CLOCK DISABLE(*)
19395 ;THIS TEST WILL CHECK THAT BCSR<12> DISABLES RESPONSE
19396 ;OF LKS REGISTER.
19397
19398 ;BCSR <12> LINE CLOCK STATUS REGISTER DISABLE
19399
19400
19401 ;ROUTINE TEST
19402
19403 ;IF UFD AND LKS IS NOT DISABLED THEN
19404 ;EXIT TEST
19405 ;ENDIF
19406
19407 ;. WRITE BCSR<12>=1
19408 ;. LET 4=ADDRESS OF LKS_TRAP
19409 ;. LET TRAP_LKS=0
19410 ;. READ LKS
19411 ;. IF TRAP_LKS NE 1 THEN
19412 ;. ERROR
19413 ;. ENDIF
19414
19415 ;ENDROUTINE
19416
19417
19418 ;ROUTINE LKS_TRAP
19419
19420 ;. LET TRAP_LKS=1
19421 ;. LET BCSR<12>=0
19422
19423 ;RTI
19424
19425
19426 ;*****
19427 117464 000004 TST37: SCOPE BIT #BIT06,#52 ;UFD MODE?
19428 117466 032737 000100 000052 BEQ 1# ;IF NOT, BRANCH
19429 117474 001404 BIT #BIT12,BCSR ;LKS DISABLED?
19430 117476 032737 010000 177520 BNE TST40 ;;IF DISABLED, EXIT TEST
19431
19432 ; CHECK BCSR<12> TO BE 0 AND 1
19433
19434 117506 013737 177520 002730 1#: MOV BCSR,SAVBR ;SAVE BCSR REGISTER
19435 117514 042737 010000 177520 BIC #BIT12,BCSR ;CLEAR BCSR
19436 117522 032737 010000 177520 BIT #BIT12,BCSR ;<12>=0?
19437 117530 001403 BEQ 2# ;IF OK, BRANCH
19438 117532 005037 001124 CLR $GDDAT ;CLEAR EXPECTED PATTERN
19439 117536 104052 ERROR +52 ;ERROR BCSR READ/WRITE BITS
19440 117540 052737 010000 177520 2#: BIS #BIT12,BCSR ;SET BIT 12
19441 117546 032737 010000 177520 BIT #BIT12,BCSR ;GOT SET OK?
19442 117554 001004 BNE 3# ;IF OK, BRANCH
19443 117556 012737 010000 001124 MOV #BIT12,$GDDAT ;EXPECTED PATTERN
19444 117564 104052 ERROR +52 ;ERROR BCSR READ/WRITE BITS
19445
19446 ; TRY TO ACCESS LKS TO GET A TIMEOUT WITH BCSR<12>=1
19447

```

TEST - LINE CLOCK DISABLE

19448	117566	013701	000004		3:	MOV	ERRVEC,R1	;SAVE TIMEOUT VECTOR
19449	117572	012737	117614	000004		MOV	#4\$,ERRVEC	;POINT TO PROGRAM AREA
19450	117600	012737	000340	000006		MOV	#340,ERRVEC+2	;AT PRIORITY 7
19451	117606	005737	177546			TST	LKS	;ACCESS LKS REGISTER
19452	117612	104066				ERROR	+66	;BCSR<12> DOES NOT DISABLE LKS
19453	117614	005726			4:	TST	(SP)+	;RESTORE STACK POINTER
19454	117616	005726				TST	(SP)+	
19455	117620	010137	000004			MOV	R1,ERRVEC	;RESTORE TIMEOUT VECTOR
19456	117624	013737	002730	177520		MOV	SAVBR,BCSR	;RESTORE BCSR
19457								
19458								

TEST - UNCONDITIONAL CLOCK LINE INTERRUPTS

```

19460 .SBTTL TEST UNCONDITIONAL CLOCK LINE INTERRUPTS
19461 ;UNCONDITIONAL CLOCK LINE INTERRUPTS(*)
19462 ;THIS TEST WILL CHECK THAT SETTING BCSR<13> TO 1 WILL
19463 ;REQUEST INTERRUPTS WHENEVER A CLOCK LINE IS ASSERTED. THIS
19464 ;SHOULD HAPPEN WITHOUT ACCESSING LKS, THEREFORE, EVEN WITH BCSR<12>=1
19465 ;INTERRUPTS SHOULD HAPPEN.
19466 ;
19467 ;BCSR <13> FORCE LINE CLOCK INTERRUPT ENABLE
19468 ;
19469 ;
19470 ;ROUTINE TEST
19471 ;IF UFD AND FORCE LKS NOT DISABLED THEN
19472 ;. EXIT TEST
19473 ;ENDIF
19474 ;. LET 100=ADDRESS OF UNCONDITIONAL_INTERRUPT_ROUTINE
19475 ;. DO FOR BCSR<12> FROM #0 TO #1(LKS DISABLED AND ENABLED)
19476 ;. (IF UFD DO ONLY FOR SELECTED LINE CLOCK)
19477 ;. . CLEAR UNCONDITIONAL_INTERRUPT
19478 ;. . SET COUNTER TO WAIT FOR 3 INTERRUPTS
19479 ;. . LET BCSR<13>=1
19480 ;. . REPEAT
19481 ;. . . DECREMENT COUNTER
19482 ;. . . UNTIL UNCONDITIONAL_INTERRUPT EQ #3 OR COUNTER EQ #0
19483 ;. . . IF COUNTER GT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
19484 ;. . . ERROR (INTERRUPTS NEVER GO LOW)
19485 ;. . . ENDIF
19486 ;. . . IF UNCONDITIONAL_INTERRUPT LT #3 THEN
19487 ;. . . ERROR (INTERRUPTS DON'T HAPPEN)
19488 ;. . . ENDIF
19489 ;. . LET BCSR<13>=#0
19490 ;. . ENDDO
19491 ;ENDROUTINE
19492 ;
19493 ;ROUTINE UNCONDITIONAL_INTERRUPT_ROUTINE
19494 ;. INCREMENT UNCONDITIONAL_INTERRUPT
19495 ;RETURN
19496 ;
19497 ;:*****
19498 117632 000004 TST40: SCOPE
19499 117634 032737 000100 000052 BIT #BIT06,#52 ;UFD MODE?
19500 117642 001404 BEQ 1# ;IF NOT, BRANCH
19501 117644 032737 020000 177520 BIT #BIT13,BCSR ;FORCE INTERRUPT SET?
19502 117652 001530 BEQ TST41 ;;IF SET, EXIT TEST
19503 ;
19504 ; CHECK BCSR<13> TO BE 0 AND 1
19505 117654 013737 177520 002730 1#: MOV BCSR,SAVBR ;SAVE BCSR REGISTER
19506 117662 042737 001000 177520 BIC #1000,BCSR ;DISABLE HALT ON BREAK
19507 117670 042737 020000 177520 BIC #BIT13,BCSR ;CLEAR BCSR
19508 117676 032737 020000 177520 BIT #BIT13,BCSR ;<13>=0?
19509 117704 001403 BEQ 2# ;IF OK, BRANCH
19510 117706 005037 001124 CLR $GDDAT ;CLEAR EXPECTED PATTERN
19511 117712 104052 ERROR +52 ;ERROR BCSR READ/WRITE BITS
19512 117714 052737 020000 177520 2#: BIS #BIT13,BCSR ;SET BIT 13
19513 117722 032737 020000 177520 BIT #BIT13,BCSR ;GOT SET OK?
19514 117730 001004 BNE 3# ;IF OK, BRANCH
19515 117732 012737 020000 001124 MOV #BIT13,$GDDAT ;EXPECTED PATTERN

```

TEST UNCONDITIONAL CLOCK LINE INTERRUPTS

```

19516 117740 104052          ERROR +52          ;ERROR BCSR READ/WRITE BITS
19517                          ;
19518                          ; SET UP TO DO UNCONDITIONAL INTERRUPTS
19519                          ;
19520 117742 012737 137062 000100 3#:  MOV  #LKSINT,8#100      ;SET UP INTERRUPT VECTOR
19521 117750 012737 000340 000102      MOV  #340,8#102      ;AT PRIORITY 7
19522 117756 052737 010000 177520      BIS  #BIT12,BCSR     ;FOR 1ST TIME DISABLE LKS
19523 117764 000403                          BR   5#              ;GO DO IT
19524 117766 042737 010000 177520 4#:  BIC  #BIT12,BCSR     ;FOR THE 2ND TIME, ENABLE LKS
19525 117774 005037 002722              5#:  CLR  LKSFL          ;CLEAR INTERRUPTS FLAG
19526 120000 012702 077777              MOV  #77777,R2      ;COUNTER TO WAIT FOR INTERRUPTS
19527 120004 106427 000240              MTPS #240           ;LOWER PRIORITY TO 5
19528 120010 022737 000003 002722 6#:  CMP  #3,LKSFL       ;3 INTERRUPTS HAPPENED?
19529 120016 001401                          BEQ  7#              ;EXIT LOOP, IF SO
19530 120020 077205                          SOB  R2,6#          ;OTHERWISE, KEEP WAITING
19531 120022 106427 000340              7#:  MTPS #340        ;RAISE PRIORITY TO 7
19532 120026 022702 077700              CMP  #77700,R2     ;INTERRUPTS HAPPEN TOO OFTEN?
19533 120032 002001                          BGE  8#              ;IF NOT, BRANCH
19534 120034 104063                          ERROR +63           ;READY LINE DOESN'T GO LOW
19535 120036 022737 000003 002722 8#:  CMP  #3,LKSFL       ;AT LEAST 3 INTERRUPTS HAPPENED?
19536 120044 002004                          BGE  9#              ;IF SO, BRANCH
19537 120046 012737 000003 001124      MOV  #3,#GDDAT     ;EXPECTED DATA
19538 120054 104064                          ERROR +64           ;INTERRUPTS DON'T HAPPEN
19539 120056 032737 010000 177520 9#:  BIT  #BIT12,BCSR     ;SECOND TIME THRU THE LOOP?
19540 120064 001340                          BNE  4#              ;IF NOT, DO IT AGAIN
19541 120066 032737 000100 000052      BIT  #BIT06,8#52   ;UFD MODE?
19542 120074 001404                          BEQ  10#             ;IF NOT, BRANCH
19543 120076 032737 010000 177520      BIT  #BIT12,BCSR   ;IF UFD AND LKS DISABLED?
19544 120104 001010                          BNE  12#             ;DON'T CHECK LKS
19545 120106 032737 000100 177546 10#: BIT  #BIT06,LKS     ;INTERRUPT ENABLE LINE HOLD 1?
19546 120114 001001                          BNE  11#             ;IF SO, BRANCH
19547 120116 104067                          ERROR +67           ;BCSR<13> DOESN'T SET LKS<6>
19548 120120 042737 000100 177546 11#: BIC  #BIT06,LKS     ;DISABLE LKS INTERRUPTS
19549 120126 013737 002730 177520 12#: MOV  SAVBR,BCSR     ;RESTORE BCSR
19550
19551

```

TEST - RESETTING LKS

```

19553 .SBTTL TEST RESETTING LKS
19554 ;RESETTING LKS(*)
19555 ;THIS TEST WILL PROVE THAT RESET INSTRUCTION SETS LKS<07> AND
19556 ;CLEARS LKS<06>.
19557 ;ROUTINE TEST
19558 ;IF UFD AND LKS IS DISABLED THEN
19559 ;. EXIT TEST
19560 ;ENDIF
19561 ;. POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL_INTERRUPT
19562 ;. SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
19563 ;. LET LKS<06>=#1
19564 ;. CLEAR LKS (CLEARS LKS<07>)
19565 ;. EXECUTE "RESET"
19566 ;. IF LKS<7> NE #1 OR LKS<6> NE #0 THEN
19567 ;. ERROR
19568 ;. ENDIF
19569 ;. IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
19570 ;. ERROR
19571 ;. ENDIF
19572 ;ENDROUTINE
19573 ;
19574 ;ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
19575 ;. FLAG_ILLEGAL_LINE_CLOCK_INTERRUPT
19576 ;RETURN
19577
19578 ;*****
TST41: SCOPE
19579 120134 000004 TST #PASS ;FIRST PASS?
19580 120136 005737 001206 BNE TST42 ;;IF NOT FIRST PASS, EXIT TEST
19581 120142 001057 BIT #BIT06,#52 ;UFD MODE?
19582 120144 032737 000100 000052 BEQ 1# ;IF NOT, BRANCH
19583 120152 001404 BIT #BIT12,BCSR ;LKS DISABLED?
19584 120154 032737 010000 177520 BEQ TST42 ;;IF DISABLED, EXIT TEST
19585
19586 ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
19587 ;
19588 120164 013737 177520 002730 1#: MOV BCSR,SAVBR ;SAVE BCSR
19589 120172 042737 010000 177520 BIC #BIT12,BCSR ;ENABLE LKS RESPONSE
19590 120200 012737 137062 000100 MOV #LKSINT,#100 ;SET UP INTERRUPT VECTOR
19591 120206 012737 000340 000102 MOV #340,#102 ;AT PROIRITY 7
19592 120214 052737 000100 177546 BIS #BIT06,LKS ;SET INTERRUPT ENABLE BIT
19593 120222 005037 002722 CLR LKSFL ;CLEAR INTERRUPTS FLAG
19594 120226 012702 077777 MOV #77777,R2 ;COUNTER TO WAIT FOR INTERRUPTS
19595 120232 106427 000240 MTPS #240 ;LOWER PRIORITY TO 5
19596 120236 022737 000003 002722 2#: CMP #3,LKSFL ;3 INTERRUPTS HAPPENED?
19597 120244 001401 BEQ 3# ;EXIT LOOP, IF SO
19598 120246 077205 SOB R2,2# ;OTHERWISE, KEEP WAITING
19599 120250 106427 000340 3#: MTPS #340 ;RAISE PRIORITY TO 7
19600 120254 000005 RESET ;EXECUTE RESET
19601 120256 032737 C00200 177546 BIT #BIT07,LKS ;READY BIT SET?
19602 120264 001001 BNE 4# ;IF SO, BRANCH
19603 120266 104070 ERROR #70 ;RESET DOESN'T SET LKS<07.
19604 120270 032737 000100 177546 4#: BIT #BIT06,LKS ;INTERRUPT ENABLE BIT CLEARED?
19605 120276 001401 BEQ TST42 ;;IF SO, EXIT TEST
19606 120300 104071 ERROR #71 ;RESET DOESN'T CLEAR LKS
19607
19608

```

TEST LINE CLOCK INTERRUPTS

```

19610 .SBITL TEST - LINE CLOCK INTERRUPTS
19611 ;LINE CLOCK INTERRUPTS(*)
19612 ;BY SETTING TO 1 LKS<06>, THIS TEST WILL CHECK FOR INTERRUPTS
19613 ;FROM BEVENT LINE AND FROM KDJ11-B 50HZ, 60HZ, 800HZ ON BOARD SIGNALS
19614 ;(THE LATTER SIGNALS WILL BE ACCESSED BY SETTING BCSR.11-10).
19615 ;
19616 ;BCSR <11> <10> CLOCK SELECT BITS 1 AND 0
19617 ;
19618 ; 0 0 EXTERNAL BEVENT LINE
19619 ; 0 1 ON-BOARD 50 HZ
19620 ; 1 0 ON-BOARD 60 HZ
19621 ; 1 1 ON-BOARD 800 HZ
19622 ;
19623 ;ROUTINE TEST
19624 ;IF UFD THEN
19625 ;. IF LKS DISABLED THEN
19626 ;. EXIT TEST
19627 ;.
19628 ;. ENDF
19629 ;. SET FLAGS TO RUN ONLY WHAT SPECIFIED IN EAPROM
19630 ;ENDIF
19631 ;. LET 100=ADDRESS OF LKS_INTERRUPT
19632 ;. DO FOR BCSR<11;10> FROM #0 TO #3
19633 ;. . LET LKS<06>=#1
19634 ;. . WAIT FOR 10 INTERRUPTS FOR EACH CLOCK
19635 ;. . STORE ACTUAL NUMBER OF INTERRUPTS FOR EACH CLOCK
19636 ;. . LET INTERRUPT_FLAG=0
19637 ;. ENDDO
19638 ;. COMPARE NUMBER OF INTERRUPTS FOR EACH CLOCK
19639 ;ENDROUTINE
19640 ;
19641 ;ROUTINE LKS_INTERRUPT
19642 ;. INCREMENT INTERRUPT_FLAG
19643 ;RETURN
19644 ;;*****
19645 TST42: SCOPE
19646 NOP ; THIS CODE ADDED FOR APT DEFAULT BREAK
; PURPOSES- TEST TIME LONGER THAN SOME APT BREAK IN
TERVALS !
19647 120302 000004 TST CCHPAS ;have done enough inclusive passes?
19648 120304 000240 BNE 99# ; not yet
19649 120306 005737 003032 NOP ; debug aid
19650 120308 000137 120564 JMP 10# ; yes skip this
19651 120310 000240 99#: NOP
19652 120312 032737 000100 000052 BIT #BIT06,#52 ;UFD MODE?
19653 120314 001411 BEQ 1# ;IF NOT, GO DO THE TEST
19654 120316 032737 010000 177520 BIT #BIT12,BCSR ;LKS IS DISABLED?
19655 120318 001123 BNE TST43 ;;IF DISABLED, EXIT TESTS
19656 120320 005002 CLR R2 ;CLEAR R2 TO SET FLAGS
19657 120322 053702 177520 BIS BCSR,R2 ;SET R2 ACCORDING TO BCSR
19658 120324 042702 171777 BIC #171777,R2 ;LEAVE ONLY BITS <11-10>
19659 ;
19660 ; SETUP DELAY VALUES FOR INTERRUPTS, IN UFD MODE ONLY FROM THE CLOCK SPECIFIED IN BCSR<11 10>
19661 ;
19662 120356 013737 177520 002730 1#: MOV BCSR,SAVBR ;STORE BCSR
19663 120358 012737 137062 000100 MOV #LKSINT,100 ;SET UP LKS VECTOR
19664 120360 012737 000340 000102 MOV #340,102 ;AT PRIORITY 7
19665 120400 012705 120602 MOV #TIMDEL,R5 ;POINTER TO DEL

```


TEST - MAINTENANCE REGISTER TEST

```

19709 .SBTTL TEST - MAINTENANCE REGISTER TEST
19710 ;MAINTENANCE REGISTER TEST
19711 ;THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
19712 ;7-4 TO BE 0010, 2-1 TO BE 10, AND READ BITS 10-08, 03, 00
19713 ;FOR FUTURE USE. THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
19714 ;MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, CPA AVAILABLE, HALT/TRAP
19715 ;OPTION, AND AC POWER OKAY.
19716 ;ROUTINE TEST
19717 ;. IF MAINT. REG. BITS <7 4> NE 0010 OR <2 1> NE 10 THEN
19718 ;. ERROR
19719 ;. ENDIF
19720 ;. READ MAINT.REG. BITS <10 08,03,00>
19721 ;ENDROUTINE
19722
19723 ;*****
19724 120612 000004 TST43: SCOPE
19725 120614 032737 174000 177750 BIT @174000,MAIREG ;UNUSED BITS ALL ZEROS?
19726 120622 001401 BEQ 1$ ;IF OK, BRANCH
19727 120624 104132 ERROR +132 ;MAINTENANCE REGISTER ERROR
19728 120626 032737 000044 177750 1$: BIT @44,MAIREG ;<5,2> SET ?
19729 120634 001001 BNE 2$ ;IF SO, BRANCH
19730 120636 104132 ERROR +132 ;MAINTENANCE REGISTER ERROR
19731 120640 032737 000322 177750 2$: BIT @322,MAIREG ;<7,6,4,1> CLEAR?
19732 120646 001401 BEQ TST44 ;;IF YES, BRANCH
19733 120650 104132 ERROR +132
19734

```

TEST - SERIAL LINE UNIT REGISTERS

```

19736 .SBTTL TEST - SERIAL LINE UNIT REGISTERS
19737 ;SERIAL LINE UNIT TEST(*)
19738 ;BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG-
19739 ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
19740 ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
19741 ;TO 1 XCSR<02>. THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
19742 ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
19743 ;PROGRAM HANGS IN THE LOOP BACK MODE.
19744 ;ROUTINE TEST
19745 ;IF UFD AND CONSOLE NOT PRESENT
19746 ;. GO TO TEST_22
19747 ;ENDIF
19748 ;. IF BCR<07> EQ #0 THEN
19749 ;. READ BCR<2-0> TO GET BAUD RATE
19750 ;. ENDF
19751 ;. LET 4=ADDRESS_OF_TIMEOUT_ROUTINE
19752 ;. DO FOR RCSR,XCSR,RBUF,XBUF
19753 ;. READ XCSR,XCSR,RBUF,XBUF
19754 ;. IF TIMEOUT_FLAG NE #0 THEN
19755 ;. ERROR
19756 ;. ENDF
19757 ;. ENDDO
19758 ;ENDROUTINE
19759 ;
19760 ;ROUTINE TIMEOUT
19761 ;. LET TIMEOUT_FLAG=#1
19762 ;ENDROUTINE
19763
19764 ;:*****
19765 120652 000004 TST44: SCOPE
19766 120654 032737 000100 000052 BIT #BIT06,#052 ;UFD MODE?
19767 120662 001406 BEQ 1# ;IF NOT, GO DO THE TEST
19768 120664 032737 000200 177524 BIT #BIT07,BCR ;IF UFD AND CONSOLE NOT PRESENT
19769 120672 001402 BEQ 1# ;NOT TRUE, DO THE TEST
19770 120674 000137 122620 JMP SLEND ;IF TRUE, SKIP ALL SLU TESTS
19771 ;
19772 ; TRY TO ACCESS SLU REGISTERS
19773 120700 013701 000004 1#: MOV ERRVEC,R1 ;SAVE TIMEOUT VECTOR
19774 120704 012737 120730 000004 MOV #3#,ERRVEC ;POINT NEW ONE TO PROGRAM AREA
19775 120712 012737 000340 000006 MOV #340,ERRVEC+2 ;AT PRIORITY 7
19776 120720 012702 177560 MOV #RCSR,R2 ;START ACCESSING WITH RCSR
19777 120724 005712 2#: TST (R2) ;ACCESS SLU REGISTER
19778 120726 000403 BR 4# ;IF NO TIMEOUT, CONTINUE
19779 120730 010237 001126 3#: MOV R2,#BDDAT ;STORE ADDRESS THAT TIMED OUT
19780 120734 104072 ERROR +72 ;TIMEOUT ACCESSING SLU REGISTER
19781 120736 022722 177566 4#: CMP #XBUF,(R2)+ ;LAST REGISTER ACCESSED?
19782 120742 103770 BLO 2# ;IF NOT, BRANCH
19783 120744 010137 000004 MOV R1,ERRVEC ;RESTORE TIMEOUT VECTOR
19784

```

TEST - XCSR BIT 7

```

19786 .SBTTL TEST - XCSR BIT 7
19787 ;CHECK THAT XCSR<07> CAN BE 0 AND 1.
19788 ;
19789 ;XCSR <07> TRANSMITTER READY
19790 ;
19791 ;ROUTINE TEST
19792 ;. WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
19793 ;. IF XCSR<07> NE #1 THEN
19794 ;. . ERROR
19795 ;. ENDF
19796 ;. LET XBUF=#NULL
19797 ;. WAIT FOR XCSR<07>=#1
19798 ;. LET XBUF=#NULL
19799 ;. IF XCSR<07> NE 0 THEN
19800 ;. . ERROR (READY DIDN'T GO LOW)
19801 ;. ENDF
19802 ;ENDROUTINE
19803
19804 ;*****
19805 120750 000004 TST45: SCOPE
19806 120752 012701 001000 MOV #1000,R1 ;COUNTER FOR ABOUT 200MICROSEC.
19807 120756 122737 000001 001220 CMPB #APTENV,#ENV ;RUNNING IN APT MODE?
19808 120764 001003 BNE 1# ;NO, GO DO TEST
19809 120766 005737 001206 TST #PASS ;FIRST PASS?
19810 120772 001017 BNE TST46 ;;IF APT AND NOT FIRST PASS, EXIT TEST
19811 120774 105737 177564 1#: TSTB XCSR ;XCSR<7> READY 1?
19812 121000 100401 BMI 2# ;IF SO, EXIT WAIT LOOP
19813 121002 077104 SOB R1,1# ;IF NOT 1, CONTINUE WAITING
19814 121004 105737 177564 2#: TSTB XCSR ;XCSR<7>=1?
19815 121010 100401 BMI 3# ;IF YES, BRANCH
19816 121012 104073 ERROR +73 ;XCSR<7> DOES NOT BECOME 1
19817 121014 012737 000000 177566 3#: MOV #NULL,XBUF ;TRY TO TRANSMIT NULL CHARACTER
19818 121022 105737 177564 TSTB XCSR ;XCSR<7>=0
19819 121026 100001 BPL TST46 ;;IF YES, EXIT TEST
19820 121030 104073 ERROR +73 ;XMIT READY DIDN'T GO LOW
19821

```

TEST - RCSR BIT 7 AND XCSR BIT 2

```

19823 .SBTTL TEST - RCSR BIT 7 AND XCSR BIT 2
19824 ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
19825 ;
19826 ;RCSR <07> RECEIVER DONE
19827 ;XCSR <02> MAINTENANCE
19828 ;
19829 ;ROUTINE TEST
19830 ;.(CHECK RCSR<07> AND XCSR<07>)
19831 ;. WAIT FOR XCSR<07>=#1
19832 ;. LET XCSR<02>=#1 (LOOP BACK MODE)
19833 ;. LET XBUF=#125
19834 ;. WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
19835 ;. IF RCSR<07> NE #1 THEN
19836 ;. . ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
19837 ;. . WORK)
19838 ;. ENDF
19839 ;. IF RBUF NE #125 THEN
19840 ;. . ERROR
19841 ;. ENDF
19842 ;. IF RCSR<07> NE #0 THEN
19843 ;. . ERROR (RCSR<07>DOES NOT GO LOW)
19844 ;. ENDF
19845 ;. LET XCSR<02>=#0
19846 ;ENDROUTINE
19847
19848 ;*****
19849 121032 000004 TST46: SCOPE
19850 121034 012701 000013 MOV #13,R1 ;COUNTER FOR ABOUT 200MICROSEC.
19851 121040 122737 000001 001220 CMPB #APTENV,#ENV ;RUNNING IN APT MODE?
19852 121046 001003 BNE 1$ ;NO, GO DO TEST
19853 121050 005737 001206 TST $PASS ;FIRST PASS?
19854 121054 001074 BNE TST47 ;;IF APT AND NOT FIRST PASS, EXIT TEST
19855 121056 105737 177564 1$: TSTB XCSR ;XCSR<7> READY 1?
19856 121062 100375 BPL 1$ ;IF NOT 1, CONTINUE WAITING
19857 121064 052737 000004 177564 2$: BIS #BIT02,XCSR ;SET LOOP BACK MODE
19858 121072 032737 000004 177564 BIT #BIT02,XCSR ;GOT SET OK?
19859 121100 001004 BNE 3$ ;IF YES, BRANCH
19860 121102 005037 177564 CLR XCSR ;RESET TO PRINT ERROR
19861 121106 104114 ERROR +114 ;XCSR<2> DOES NOT BECOME 1
19862 121110 000456 BR TST47 ;;EXIT TEST
19863 ;
19864 ; STALL FOR A WHILE IN CASE XCSR<2> CAUSES RCSR<7> TO BE 1
19865 121112 012701 060000 3$: MOV #60000,R1 ;STALL IN CASE XCSR<2> SETS READY
19866 121116 105737 177560 4$: TSTB RCSR ;IF RECEIVER READY SET?
19867 121122 100402 BMI 5$ ;IF SET, BRANCH
19868 121124 077104 SOB R1,4$ ;OTHERWISE, STAY FOR A WHILE
19869 121126 000402 BR 6$ ;IF NOT READY, BRANCH
19870 121130 005737 177562 5$: TST RBUF ;READ RBUF
19871 ;
19872 ; TRANSMIT XON AND CHECK RCSR<7>
19873 ;
19874 121134 012737 000021 177566 6$: MOV #21,XBUF ;TRANSMIT A CHARACTER
19875 121142 012701 060000 MOV #60000,R1 ;COUNTER TO WAIT
19876 121146 105737 177560 7$: TSTB RCSR ;RCSR<7> READY 1?
19877 121152 100401 BMI 8$ ;IF YES, EXIT WAIT LOOP
19878 121154 077104 SOB R1,7$ ;OTHERWISE, CONTINUE WAITING

```


TEST - RESET AND XCSR<2!0>

```

19897 .SBTTL TEST - RESET AND XCSR<2!0>
19898 ;CHECK THAT RESET CLEARS XCSR<0!2>.
19899 ;ROUTINE TEST
19900 ;.(CHECK RCSR<07> AND XCSR<07> AND RESET)
19901 ;. LET XCSR<02,00>=#1 (LOOP BACK MODE)
19902 ;. EXECUTE "RESET"
19903 ;. IF XCSR<02!00> NE #0 THEN
19904 ;. ERROR
19905 ;. ENDIF
19906 ;. LET XCSR<02>=#0
19907 ;ENDROUTINE
19908
19909 ;:*****
19910 121246 000004 TST47: SCOPE
19911 121250 005737 001206 TST $PASS ;FIRST PASS?
19912 121254 001011 BNE TST50 ;;IF NOT FIRST PASS, EXIT TEST
19913 121256 052737 000005 177564 1$: BIS #BIT02!BIT00,XCSR ;LOOP BACK MODE
19914 ;
19915 ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
19916 ;
19917 121264 000005 RESET ;EXECUTE RESET
19918 121266 032737 000005 177564 BIT #BIT02!BIT00,XCSR ;XCSR<2,0> CLEAR?
19919 121274 001401 BEQ TST50 ;;IF YES, BRANCH
19920 121276 104102 ERROR +102 ;XCSR<2,0> NOT CLEARED ON RESET
19921

```

TEST - RESET AND INTERRUPT ENABLE BITS

```

19923 .SBTTL TEST RESET AND INTERRUPT ENABLE BITS
19924 ;CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
19925 ;CLEARS XCSR<06> AND RCSR<06>.
19926 ;
19927 ;RCSR <06> RECEIVER INTERRUPT ENABLE
19928 ;XCSR <06> TRANSMITTER INTERRUPT ENABLE
19929 ;
19930 ;ROUTINE TEST
19931 ;. LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT_XCSR
19932 ;. LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT_XCSR
19933 ;. SET PRIORITY TO 4
19934 ;. LET XCSR<02>=#1 (LOOPBACK MODE)
19935 ;. LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
19936 ;. LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
19937 ;. WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
19938 ;. LET XBUF=#NULL (SEND A CHARACTER)
19939 ;. WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
19940 ;. EXECUTE "RESET"
19941 ;. IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XCSR NE #0 THEN
19942 ;. ERROR
19943 ;. ENDIF
19944 ;. RESTORE PRIORITY TO NORMAL
19945 ;ENDROUTINE
19946 ;
19947 ;ROUTINE ILLEGAL_INTERRUPT_XCSR
19948 ;. INCREMENT XCSR
19949 ;ENDROUTINE
19950 ;
19951 ;*****
19952 121300 000004 TST50: SCOPE
19953 121302 005737 001206 TST $PASS ;FIRST PASS?
19954 121306 001033 BNE 5$ ;SKIP RESET PART OF THE TEST
19955 ;
19956 ; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRASMITTER OF SLU
19957 ; ARE CLEARED BY RESET
19958 121310 052737 000100 177564 1$: BIS #BIT06,XCSR ;SET INTERRUPT ENABLE BIT IN XCSR
19959 121316 032737 000100 177564 BIT #BIT06,XCSR ;GOT SET OK?
19960 121324 001001 BNE 2$ ;IF YES, BRANCH
19961 121326 104110 ERROR +110 ;IN BIT 6 OF XCSR
19962 121330 052737 000100 177560 2$: BIS #BIT06,RCSR ;SET INTERRUPT ENABLE BIT IN RCSR
19963 121336 032737 000100 177560 BIT #BIT06,RCSR ;GOT SET OK?
19964 121344 001001 BNE 3$ ;IF YES, BRANCH
19965 121346 104110 ERROR +110 ;IN BIT 6 OF RCSR
19966 121350 000005 3$: RESET ;INLINE BUS RESET
19967 121352 032737 000100 177564 BIT #BIT06,XCSR ;XMIT INTERRUPT ENABLE BIT CLEARED?
19968 121360 001401 BEQ 4$ ;IF CLEARED, BRANCH
19969 121362 104102 ERROR +102 ;INTERRUPT ENABLE NOT CLEARED ON RESET
19970 121364 032737 000100 177560 4$: BIT #BIT06,RCSR ;RECEIVE INTERRUPT ENBLE CLEARED?
19971 121372 001401 BEQ 5$ ;IF CLEARED, BRANCH
19972 121374 104102 ERROR +102 ;INTERRUPT ENABLE NOT CLEARED ON RESET
19973 ;
19974 ; CHECK THAT TRANSMIT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
19975 ;
19976 121376 012737 121444 000064 5$: MOV #9$,#64 ;POINT XMIT VECTOR TO PROGRAM AREA
19977 121404 012737 000340 000066 MOV #340,#66 ;AT PRIORITY 7
19978 121412 052737 000100 177564 BIS #BIT06,XCSR ;SET INTERRUPT ENABLE BIT IN XCSR

```


TEST - RESET AND INTERRUPT ENABLE BITS

```

19979 121420 012702 000340      MOV      #340,R2      ;SET PRIORITY TO 7
19980 121424 000402              BR        7#          ;GO WAIT IN CASE OF INTERRUPTS
19981 121426 162702 000040      6#:     SUB      #40,R2      ;LOWER PRIORITY LEVEL
19982 121432 106402              7#:     MTPS     R2          ;SET PRIORITY
19983 121434 012703 000026      MOV      #26,R3      ;TIME DELAY
19984 121440 077301              8#:     SOB      R3,8#      ;WAIT FOR INTERRUPTS
19985 121442 000403              BR        10#         ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
19986 121444 104101              9#:     ERROR   +101       ;INTERRUPTS HAPPEN AT WRONG PRIORITY
19987 121446 005726              TST      (SP)+        ;CLEAN UP THE STACK
19988 121450 005726              TST      (SP)+
19989 121452 022702 000200      10#:    CMP      #200,R2     ;AT PRIORITY 4?
19990 121456 001363              BNE      6#           ;IF NOT LAST ONE, CONTINUE
19991 121460 106427 000340      MTPS     #340        ;RESTORE PRIORITY 7
19992 121464 042737 000100 177564 BIC      #BIT06,XCSR   ;CLEAR INTERRUPT ENABLE BIT
19993
19994 ; CHECK THAT RECEIVE INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
19995 ;
19996 121472 012737 121562 000060      MOV      #15#,R6     ;POINT RECEIVE VECTOR TO PROGRAM AREA
19997 121500 012737 000340 000062      MOV      #340,R6     ;AT PRIORITY 7
19998 121506 105737 177564              11#:    TSTB     XCSR        ;TRANSMITTER READY
19999 121512 100375              BPL      11#         ;IF NOT, WAIT
20000 121514 012737 000000 177566      MOV      #NULL,XBUF   ;TRY TO TRANSMIT NULL
20001 121522 052737 000100 177560      BIS      #BIT06,RCSR  ;SET INTERRUPT ENABLE BIT IN RCSR
20002 121530 012702 000340      MOV      #340,R2     ;SET PRIORITY TO 7
20003 121534 000402              BR        13#         ;GO WAIT IN CASE OF INTERRUPTS
20004 121536 162702 000040      12#:    SUB      #40,R2     ;LOWER PRIORITY LEVEL
20005 121542 052737 000004 177564 13#:    BIS      #BIT02,XCSR  ;SET LOOP BACK MODE
20006 121550 106402              MTPS     R2          ;SET PRIORITY
20007 121552 012703 000100      MOV      #100,R3     ;TIME DELAY
20008 121556 077301              14#:    SOB      R3,14#     ;WAIT FOR INTERRUPTS
20009 121560 000406              BR        16#         ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
20010 121562 042737 000004 177564 15#:    BIC      #BIT02,XCSR  ;CLEAR LOOP BACK MODE
20011 121570 104101              ERROR   +101       ;INTERRUPTS HAPPEN AT WRONG PRIORITY
20012 121572 005726              TST      (SP)+        ;CLEAN UP THE STACK
20013 121574 005726              TST      (SP)+
20014 121576 022702 000200      16#:    CMP      #200,R2     ;AT PRIORITY 4?
20015 121602 001355              BNE      12#         ;IF NOT LAST ONE, CONTINUE
20016
20017 ; CLEAN UP BEFORE NEXT TEST
20018 ;
20019 121604 106427 000340      MTPS     #340        ;RESTORE PRIORITY 7
20020 121610 042737 000100 177560      BIC      #BIT06,RCSR  ;CLEAR INTERRUPT ENABLE BIT
20021 121616 012702 000300      MOV      #300,R2     ;STALL DELAY
20022 121622 105737 177560              17#:    TSTB     RCSR        ;RECEIVE READY?
20023 121626 100401              BMI      18#         ;STOP WAITING, IF SO
20024 121630 077204              SOB      R2,17#     ;OTHERWISE, STAY IN THE LOOP
20025 121632 005737 177562 18#:    TST      RBUF        ;READ CHARACTER TRANSMITTED
20026 121636 042737 000004 177564      BIC      #BIT02,XCSR  ;CLEAR LOOP BACK MODE
20027

```

TEST - INTERRUPT PRIORITY FOR SLU

```

20029 .SBTTL TEST - INTERRUPT PRIORITY FOR SLU
20030 ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
20031 ;RCSR<06> AND XCSR<06>.
20032 ;
20033 ;ROUTINE TEST
20034 ;. LET 60=ADDRESS_OF_LEGAL_RINTERRUPT
20035 ;. LET 64=ADDRESS_OF_LEGAL_XINTERRUPT
20036 ;. LET XCSR<02>=#1
20037 ;. SET PRIORITY TO #3
20038 ;. WAIT FOR XINTERRUPT=#3
20039 ;. IF XCSR<07> EQ #1 THEN
20040 ;. ERROR
20041 ;. ENDF
20042 ;. WAIT FOR RINTERRUPT=#3
20043 ;. IF RCSR<07> EQ #0 THEN
20044 ;. ERROR
20045 ;. ENDF
20046 ;. LET XCSR<02>=#0
20047 ;. SET PRIORITY TO NORMAL
20048 ;ENDROUTINE
20049 ;
20050 ;ROUTINE LEGAL_XINTERRUPT
20051 ;. LET XBUF=#CHARACTER
20052 ;. INCREMENT XINTERRUPT
20053 ;ENDROUTINE
20054 ;
20055 ;ROUTINE LEGAL_RINTERRUPT
20056 ;. READ RCSR
20057 ;. INCREMENT RINTERRUPT
20058 ;ENDROUTINE
20059 ;
20060 ;*****
20061 121644 000004 TST51: SCOPE
20062 121646 122737 000001 001220 CMPB #APTENV,#ENV ;RUNNING IN APT MODE?
20063 121654 001003 BNE 100# ;NO. GO DO TEST
20064 121656 005737 001206 TST #PASS ;FIRST PASS?
20065 121662 001113 BNE TST52 ;;IF APT AND NOT FIRST PASS, EXIT TEST
20066 ;
20067 ; GET READY FOR INTERRUPTS
20068 121664 012737 122060 000060 100#: MOV #8,#060 ;STORE RECEIVER VECTOR
20069 121672 012737 122004 000064 MOV #6,#064 ;STORE TRANSMITTER VECTOR
20070 121700 012737 000340 000062 MOV #340,#062 ;AT PRIORITY 7
20071 121706 012737 000340 000066 MOV #340,#066 ;FOR RECEIVER AND TRANSMITTER
20072 121714 052737 000004 177564 BIS #BIT02,XCSR ;SET LOOP BACK MODE
20073 121722 012701 000100 MOV #100,R1 ;DELAY FOR UNEXPECTED CHARACTERS
20074 121726 105737 177560 1#: TSTB RCSR ;RECEIVER READY?
20075 121732 100401 BMI 2# ;IF YES, BRANCH
20076 121734 077104 SOB R1,1# ;OTHERWISE, WAIT JUST IN CASE
20077 121736 005737 177562 2#: TST RBUF ;READ RECEIVER
20078 ;
20079 ; SET PRIORITIES AND XMIT INTERRUPTS
20080 ;
20081 121742 012702 000140 MOV #140,R2 ;START WITH PRIORITY 3
20082 121746 000402 BR 4# ;TRY TO DO IT
20083 121750 162702 000040 3#: SUB #40,R2 ;LOWER PRIORITY
20084 121754 106402 4#: MTPS R2 ;TRY TO DO AT LOWER PRIORITY

```

TEST - INTERRUPT PRIORITY FOR SLU

```

20085 121756 052737 000100 177564      BIS      #BIT06,XCSR      ;LOOP BACK & INTERRUPT ENABLE
20086 121764 012703 001000              MOV      #1000,R3      ;WAIT DELAY FOR INTERRUPTS
20087 121770 077301              SOB      R3,5#        ;WAIT FOR XMIT INTERRUPTS
20088 121772 042737 000004 177564      BIC      #BIT02,X^R    ;CLEAR LOOP BACK BIT
20089 122000 104107              ERROR   +107          ;NO XMIT INTERRUPTS
20090 122002 000443              BR      TST52         ;;IF ERROR, EXIT TEST
20091              ;
20092              ; TRANSMITTER INTERRUPT HERE
20093              ;
20094 122004 005726              6#:     TST      (SP)+   ;CLEAN UP STACK
20095 122006 005726              TST      (SP)+
20096 122010 042737 000100 177564      BIC      #BIT06,XCSR   ;CLEAR INTERRUPT ENABLE
20097 122016 012737 000000 177566      MOV      #NULL,XBUF   ;TRANSMIT NULL
20098 122024 052737 000100 177560      BIS      #BIT06,RCSR   ;SET RECEIVE INTERRUPT
20099 122032 106402              MTPS    R2            ;SET NEXT PRIORITY
20100 122034 012703 100000              MOV      #100000,R3   ;WAIT DELAY FOR INTERRUPTS
20101 122040 077301              7#:     SOB      R3,7#   ;WAIT FOR RECEIVE INTERUPTS
20102 122042 042737 000004 177564      BIC      #BIT02,XCSR   ;CLEAR LOOP BACK MODE BIT
20103 122050 104107              ERROR   +107          ;NO RECEIVE INTERRUPTS
20104 122052 000406              BR      9#            ;DON'T TOUCH STACK
20105 122054 106427 000340              MTPS    #340         ;RAISE PRIORITY
20106              ;
20107              ; RECEIVER INTERRUPT HERE
20108              ;
20109 122060 005726              8#:     TST      (SP)+   ;CLEAN UP STACK
20110 122062 005726              TST      (SP)+
20111 122064 005737 177562              TST      RBUF         ;READ RECEIVER BUFFER
20112 122070 005702              9#:     TST      R2            ;PRIORITY 0
20113 122072 001326              BNE     3#            ;IF NOT YET, CONTINUE
20114 122074 106427 000340              MTPS    #340         ;RAISE PRIORITY TO 7
20115 122100 042737 000100 177560      BIC      #BIT06,RCSR   ;CLEAR RECEIVE INTER. ENABLE
20116 122106 005037 177564              CLR     XCSR         ;CLEAR XCSR
20117

```

TEST - BREAK CONDITION

```

20119 .SBTTL TEST BREAK CONDITION
20120 ;CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
20121 ;
20122 ;RCSR <15> ERROR
20123 ; <13> FRAMING ERROR
20124 ; <11> RECEIVED BREAK
20125 ;
20126 ;XCSR <00> TRANSMIT BREAK
20127 ;
20128 ;ROUTINE TEST
20129 ;. LET XCSR<02>=#1
20130 ;. LET XCSR<00>=#1
20131 ;. WAIT FOR RCSR<07>=#1
20132 ;. IF RBUF<15!13!11> NE #1 THEN
20133 ;. ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
20134 ;. ENDF
20135 ;. LET XCSR<00>=#0
20136 ;. IF XCSR<00> NE #0 THEN
20137 ;. ERROR (XCSR<00> DOES NOT GO LOW)
20138 ;. ENDF
20139 ;. WAIT FOR XCSR<07>=#1
20140 ;. LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
20141 ;. WAIT FOR RCSR<07>=#1
20142 ;. IF RBUF<15!13!11> NE #0 THEN
20143 ;. ERROR
20144 ;. ENDF
20145 ;. LET XCSR<00>=#1
20146 ;. EXECUTE "RESET"
20147 ;. IF XCSR<00> NE #0 THEN
20148 ;. ERROR
20149 ;. ENDF
20150 ;. LET XCSR<02>=#0
20151 ;ENDROUTINE
20152
20153 ;*****
122112 000004 TST52: SCOPE
20154 ;
20155 ; DECIDE WHETHER TO RUN THIS TEST
20156 ;
20157 122114 032737 000200 000052 BIT #BIT07,#52 ;UFD MODE?
20158 122122 001127 BNE TST53 ;;IN UFD MODE, EXIT TEST
20159 122124 005737 001206 TST #PASS ;FIRST PASS?
20160 122130 001124 BNE TST53 ;;IF APT AND NOT FIRST PASS, EXIT TEST
20161 ;
20162 ; SEND BREAK AND CHECK ERROR BITS IN RBUF
20163 ;
20164 122132 052737 000004 177564 1#: BIS #BIT02,XCSR ;TRANSMIT IN LOOP BACK
20165 122140 013737 177520 002730 MOV BCSR,SAVBR ;SAVE BCSR
20166 122146 042737 001000 177520 BIC #BIT09,BCSR ;DISABLE HALT ON BREAK
20167 122154 052737 000001 177564 BIS #BIT00,XCSR ;SET SEND BREAK BIT
20168 122162 032737 000001 177564 BIT #BIT00,XCSR ;GOT SET OK?
20169 122170 001001 BNE 2# ;IF YES, BRANCH
20170 122172 104110 ERROR -110 ;WRITING 1 TO XCSR<0>
20171 122174 012701 000100 2#: MOV #100,R1 ;STALL DELAY
20172 122200 105737 177560 4#: TSTB RCSR ;RECEIVER READY?
20173 122204 100401 BMI 5# ;IF YES, BRANCH
20174 122206 077104 SOB R1,4# ;WAIT JUST IN CASE OF A CHARACTER

```

TEST - BREAK CONDITION

```

20175 122210 005737 177562      5:  TST  RBUF          ;READ A CHARACTER
20176 122214 052737 000001 177564  BIS  #BIT00,XCSR   ;TRANSMIT BREAK
20177 122222 012701 001000      MOV  #1000,R1     ;ANOTHER DELAY TO GET BREAK
20178 122226 077101      SOB  R1,6#        ;WAIT A WHILE
20179 122230 105737 177560      6:  TSTB RCSR       ;RECEIVER READY?
20180 122234 100375      7:  BPL  7#          ;IF NOT, WAIT
20181 122236 013737 177562 001126  MOV  RBUF,#BDDAT  ;STORE WHATEVER RECEIVED
20182 122244 022737 124000 001126  CMP  #BIT15!BIT13!BIT11,#BDDAT ;ALL ERROR BITS SET?
20183 122252 001405      BEQ  8#           ;IF YES, BRANCH
20184 122254 042737 000004 177564  BIC  #BIT02,XCSR  ;RESET TO ENABLE SLU
20185 122262 104105      ERROR +105       ;BREAK DOES NOT CAUSE ERRORS
20186 122264 000446      BR   TST53        ;;EXIT
20187 122266 042737 000001 177564  8:  BIC  #BIT00,XCSR ;CLEAR TRANSMIT BREAK
20188 122274 032737 000001 177564  BIT  #BIT00,XCSR  ;GET CLEARED OK?
20189 122302 001405      BEQ  9#           ;IF YES, BRANCH
20190 122304 042737 000004 177564  BIC  #BIT02,XCSR  ;RESET TO ENABLE SLU
20191 122312 104110      ERROR +110       ;ERROR WRITING 0 TO XCSR<0>
20192 122314 000432      BR   TST53        ;;EXIT
20193
20194      ; CHECK THAT BREAK CONDITION IS CLEARED
20195
20196 122316 013737 002730 177520  9:  MOV  SAVBR,BCSR ;RESTORE BCSR
20197 122324 105737 177564      10: TSTB XCSR       ;XMIT READY?
20198 122330 100375      BPL  10#         ;IF NOT, WAIT
20199 122332 012737 000177 177566  MOV  #177,XBUF   ;TRY TO TRANSMIT DELETE
20200 122340 105737 177560      11: TSTB RCSR       ;RECEIVER READY
20201 122344 100375      BPL  11#         ;IF NOT, WAIT
20202 122346 013737 177562 001126  MOV  RBUF,#BDDAT ;STORE RECEIVE BUFFER
20203 122354 032737 124000 001126  BIT  #BIT15!BIT13!BIT11,#BDDAT ;ERRORS CLEARED?
20204 122362 001404      BEQ  12#         ;IF YES, BRANCH
20205 122364 042737 000004 177564  BIC  #BIT02,XCSR  ;RESET TO ENABLE SLU
20206 122372 104106      ERROR +106       ;BREAK NOT CLEARED ON NEXT CHARACTER
20207 122374 042737 000004 177564  12: BIC  #BIT02,XCSR ;CLEAR LOOP BACK MODE
20208
20209

```

TEST - OVERRUN CONDITION

```

20211 .SBTTL TEST OVERRUN CONDITION
20212 ;CHECK OVERRUN CONDITION
20213 ;
20214 ;RCSR <14> OVERRUN ERROR
20215 ;
20216 ;ROUTINE TEST
20217 ;. LET XCSR<02>=#1 (LOOPBACK MODE)
20218 ;. WAIT FOR XCSR<07>=#1
20219 ;. LET XBUF=#252
20220 ;. WAIT FOR XCSR<07>=#1
20221 ;. LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
20222 ;. WAIT FOR RCSR<07>=#1
20223 ;. STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
20224 ;. IF LOW BYTE OF RBUF NE #125 THEN
20225 ;. . ERROR (1ST CHARACTER WASN'T OVERRUN)
20226 ;. ENDF
20227 ;. IF RBUF<15!14> NE #1 THEN
20228 ;. . ERROR (NO OVERRUN BIT SET)
20229 ;. ENDF
20230 ;. WAIT FOR XCSR<07>=#1
20231 ;. LET XBUF=#NULL
20232 ;. WAIT FOR RCSR<07>=#1
20233 ;. IF RBUF<15!14> NE #0 THEN
20234 ;. . ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
20235 ;. ENDF
20236 ;. LET XCSR<02>=#0
20237 ;ENDROUTINE
20238
20239 ;*****
TST53: SCOPE
20240 122402 000004 CMPB #APTENV,$ENV ;RUNNING IN APT MODE?
20241 122404 122737 000001 001220 BNE 100$ ;NO, GO DO TEST
20242 122412 001003 TST #PASS ;FIRST PASS?
20243 122414 005737 001206 BNE TST54 ;;IF APT AND NOT FIRST PASS, EXIT TEST
20244 122420 001077 BIS #BIT02,XCSR ;SET LOOP BACK MODE
20245 122430 052737 000004 177564 100$: TSTB XCSR ;READY TO TRANSMIT?
20246 122434 100375 BPL 1$ ;IF NOT, WAIT
20247 122436 012737 000021 177566 MOV #21,XBUF ;TRANSMIT A CHARACTER
20248 122444 105737 177560 2$: TSTB RCSR ;RECEIVE READY?
20249 122450 100375 BPL 2$ ;IF NOT, WAIT
20250 122452 105737 177564 3$: TSTB XCSR ;READY TO TRANSMIT?
20251 122456 100375 BPL 3$ ;IF NOT, WAIT
20252 122460 012737 000177 177566 MOV #177,XBUF ;TRANSMIT THE 2ND CHARACTER
20253 122466 012703 175000 MOV #175000,R3 ;STALL FOR THE 2ND CHARACTER $$$
20254 122472 077301 4$: SOB R3,4$ ;WAIT A WHILE
20255 122474 013737 177562 001126 MOV RBUF,$BDDAT ;STORE RECEIVED DATA
20256 122502 012737 140177 001124 MOV #140177,$GDDAT ;EXPETED PATTERN
20257 122510 122737 000177 001126 CMPB #177,$BDDAT ;2ND CHARACTER RECEIVED?
20258 122516 001404 BEQ 5$ ;IF YES, BRANCH
20259 122520 042737 000004 177564 BJC #BIT02,XCSR ;RESET TO ENABLE SLU
20260 122526 104111 ERROR +111 ;2ND CHARACTER DIDN'T OVERRUN 1ST
20261 122530 122737 000300 001127 5$: CMPB #BIT7!BIT6,$BDDAT+1 ;OVERRUN ERROR BITS SET?
20262 122536 001404 BEQ 6$ ;IF YES, BRANCH
20263 122540 005037 177564 CLR XCSR ;RESET TO ENABLE SLU
20264 122544 104112 ERROR +112 ;OVERRUN DOES NOT SET ERRORS BITS
20265 122546 000424 BR TST54 ;;EXIT
20266 ;

```

TEST OVERRUN CONDITION

```

20267 ; SEND NEXT CHARACTER TO CLEAR OVERRUN CONDITIONS
20268 ;
20269 122550 105737 177564 6$: TSTB XCSR ;TRANSMITTER READY?
20270 122554 100375 BPL 6$ ;IF NOT, BRANCH AND WAIT
20271 122556 012737 000000 177566 MOV @NULL,XBUF ;TRANSMIT MULTI CHARACTER
20272 122564 105737 177560 7$: TSTB RCSR ;RECEIVER READY?
20273 122570 100375 BPL 7$ ;IF NOT, BRANCH AND WAIT
20274 122572 032737 140000 177562 BIT @BIT15!BIT14,RBUF ;ANY ERRORS SET?
20275 122600 001404 BEQ 8$ ;IF NOT, BRANCH
20276 122602 042737 000004 177564 BIC @BIT02,XCSR ;RESET TO ENABLE SLU
20277 122610 104113 ERROR +113 ;OVERRUN NOT CLEARED ON NEXT CHAR.
20278 122612 042737 000004 177564 8$: BIC @BIT02,XCSR ;CLEAR LOOP BACK MODE BIT
20279
20280 122620 SLEND: ;LAST SLU TEST
    
```

TEST LED'S ON

20282
20283
20284
20285
20286
20287
20288
20289
20290
20291
20292
20293
20294

```
.SBTTL TEST - LED'S ON
;LED'S ON
;THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
;DISPLAYED IN LED'S.
;
;ROUTINE TEST
;.. WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
;.. STALL ALLOWING TIME TO SEE PATTERN
;.. ROTATE LEFT TO LIGHT UP NEXT LED'S
;.. ENDDO
;ENDROUTINE
```

20295 122620 000004
20296 122622 052737 001000 177520
20297 122630 005005
20298 122632 032737 000001 000052
20299 122640 001427
20300 122642 005737 001206
20301 122646 001024
20302 122650 122737 000001 001220
20303 122656 001420
20304 122660 005105
20305 122662 104401 001175
20306 122666 104401 123016
20307 122672 012737 122770 000060
20308 122700 012737 000340 000062
20309 122706 052737 000100 177560
20310 122714 106427 000140
20311 122720 012704 000006
20312 122724 012701 000076
20313 122730 110137 177524
20314 122734 012703 000004
20315 122740 012702 177777
20316 122744 077201
20317 122746 077304
20318 122750 000261
20319 122752 006101
20320 122754 077413
20321 122756 005705
20322 122760 001407
20323 122762 104401 001170
20324 122766 000754
20325 122770 005737 177562
20326 122774 062706 000004
20327 123000 112737 000377 177524
20328 123006 042737 001000 177520
20329
20330 123016 012 015 124
123021 110 111 123
123024 040 111 123
123027 040 101 040
123032 124 105 123
123035 124 040 106
123040 117 122 040
123043 117 116 055

```
*****
TST54: SCOPE
      BIS      #1000,BCSR      ;ENABLE HOB FOR APT
      CLR      R5              ;FLAG IN NO INTERRUPT MODE
      BIT      #BIT00,#52     ;IF RUNNING IN CHAIN MODE
      BEQ      1$             ;SKIP PRINTOUTS
      TST      $PASS          ;1ST PASS?
      BNE      1$             ;IF NOT, SKIP PRINTOUTS
      CMPB     #APTENV,$ENV    ;APT MODE?
      BEQ      1$             ;YES, SKIP PRINTOUT'S
      COM      R5              ;CLEAR FLAG IN INTERRUPT MODE
      TYPE     ,#CRLF
      TYPE     ,LEDS          ;IDENTIFY THE TEST
      MOV      #5,$#60        ;RECEIVE SLU VECTOR
      MOV      #340,$#62      ;AT PRIORITY 7
      BIS      #BIT06,RCSR    ;ENABLE INTERRUPTS
      MTPS     #140           ;LOWER PRIORITY
      1$: MOV      #6,R4        ;FOR EACH LOOP
      MOV      #76,R1         ;START WITH 1
      2$: MOVB   R1,BDR        ;TURN OFF FIRST LED
      MOV      #4,R3          ;STALL DELAY
      3$: MOV      #177777,R2  ;STALL DELAY
      4$: SOB     R2,4$        ;WAIT A WHILE
      SOB     R3,3$          ;WAIT A WHILE
      SEC
      ROL      R1              ;SET CARRY
      SOB     R4,2$           ;GET ANOTHER LED
      TST      R5              ;DO A FEW TIMES
      BEQ      6$             ;RUNNING IN INTERACTIVE MODE?
      TYPE     ,#BELL         ;IF NOT, EXIT
      BR      1$              ;
      5$: TST      RBUF        ;REPEAT PATTERN
      ADD      #4,$P           ;READ BUFFER
      MOVB    #377,BDR        ;ADJUST STACK
      BIC     #1000,BCSR     ;NO MORE
      BR      TST55          ;DISABLE HOB FOR APT
      6$:
      ;;EXIT TEST
```

LEDS: .ASCII <12><15>/THIS IS A TEST FOR ON-BOARD LED'S/<12><15>

TEST - LED'S ON

	123046	102	117	101
	123051	122	104	040
	123054	114	105	104
	123057	047	123	012
	123062	015		
20331	123063	124	131	120
	123066	105	040	101
	123071	116	131	040
	123074	103	110	101
	123077	122	101	103
	123102	124	105	122
	123105	040	117	116
	123110	040	101	040
	123113	113	105	131
	123116	102	117	101
	123121	122	104	040
	123124	124	117	040
	123127	103	117	116
	123132	124	111	116
	123135	125	105	012
	123140	015	000	

.ASCIZ /TYPE ANY CHARACTER ON A KEYBOARD TO CONTINUE/<12><15>

20332
20333

.EVEN

TEST - MEMORY MAPPING

```

20335 .SBTTL TEST - MEMORY MAPPING
20336 ;MEMORY MAPPING
20337 ;THIS TEST WILL AUTOSIZE MEMORY IN 2K BYTES. EVERY PAGE WILL BE
20338 ;CHECKED FOR WHAT TYPE IT IS: Q-BUS, UNIBUS OR PMI BUS MEMORY BY
20339 ;SEEING HOW IT CAN BE CACHED.
20340 ;ROUTINE TEST
20341 ;. LET 4=ADDRESS OF NON-EXISTENT MEMORY
20342 ;. IF KMCR<05-00> NE #1 THEN (NOT ALL UNIBUS MEMORY)
20343 ;.   TURN ON MMU AND REMAP THE PROGRAM AREA
20344 ;.   REPEAT
20345 ;.     DO FOR KDPARO FROM #0 TO #177600
20346 ;.       DO FOR R1 FROM #0 TO #20000 BY #4000
20347 ;.         READ (R1)
20348 ;.         IF ABORT_NON-EXISTENT MEMORY NE 1
20349 ;.           MEMORY EXISTS
20350 ;.           READ (R1)+
20351 ;.           READ (R1)
20352 ;.           IF HIT/MISS EQ 2_HITS THEN
20353 ;.             PMI MEMORY
20354 ;.           ELSE
20355 ;.             IF HIT/MISS EQ HIT THEN
20356 ;.               Q-BUS MEMORY
20357 ;.             ELSE
20358 ;.               ERROR
20359 ;.             ENDIF
20360 ;.           ENDIF
20361 ;.         ENDIF
20362 ;.       LET ABORT_NON-EXISTENT_MEMORY=#0
20363 ;.     ENDDO
20364 ;.   ENDDO
20365 ;. UNTILL ABORT_NON-EXISTANT_MEMORY EQ #1
20366 ;. ENDIF
20367 ;.
20368 ;ENDROUTINE
20369 ;
20370 ;ROUTINE NON-EXISTENT_MEMORY
20371 ;. LET ABORT_NON-EXISTENT_MEMORY=#1
20372 ;. RETURN
20373 ;ENDROUTINE
20374 ;
20375 ;*****
TST55: SCOPE
        BIT      #BIT00,#52          ;CHAIN MODE?
        BEQ      TST56              ;;IF SO, EXIT TEST
        TST      #PASS              ;FIRST PASS?
        BNE      TST56              ;;IF APT AND NOT FIRST PASS, EXIT TEST
        CMPB    #APTENV,#ENV        ;APT MODE?
        BEQ      TST56              ;;YES, SKIP PRINTOUT'S
;
; SETUP ALL REGISTERS FOR MAPPING
;
20385 123172 012737 000400 177746      MOV      #400,CCR          ;FLUSH THE CACHE
20386 123200 013704 000004              MOV      ERRVEC,R4        ;STORE TIMEOUT VECTOR
20387 123204 012737 123354 000004      MOV      #7#,#ERRVEC     ;POINT TO PROGRAM AREA
20388 123212 012737 000340 000006      MOV      #340,ERRVEC+2   ;AT PRIORITY 4
20389 123220 004737 13657A              JSR      PC,INITMM        ;REMAP PROGRAM AREA
20390 123224 005037 172354              CLR      KIPAR6          ;USED FOR MAPPING MEMORY

```

TEST MEMORY MAPPING

```

20391 123230 005002          CLR      R2          ;CLEAR PAGE COUNT FOR PMI
20392 123232 005003          CLR      R3          ;CLEAR PAGE COUNT FOR QBUS
20393 123234 005237 177572   INC      MMRO        ;ENABLE MEMORY MANGEMENT
20394 123240 052737 000020 172516   BIS      @BIT04,MMR3 ;ENABLE 22 BITS
20395 123246 000403          BR       2$         ;GO ACCESS
20396
20397          ;
20398          ; TRY TO MAP ALL PAGES
20399 123250 062737 000200 172354 1$:      ADD      @200,KIPAR6 ;INCREMENT BY 4K WORDS
20400 123256 012701 140000 2$:      MOV      @140000,R1 ;ACCESS THRU KIPAR6
20401 123262 000402          BR       4$         ;START DOING IT
20402 123264 062701 003776 3$:      ADD      @3776,R1   ;INCREMENT BY 1K WORDS
20403 123270 005721 4$:      TST      (R1)+    ;ACCESS 1ST LOCATION
20404 123272 042737 001000 177520   BIC      @1000,BCSR ;DISABLE HALT ON BREAK
20405 123300 005711          TST      (R1)      ;ACCESS 2ND LOCATION
20406 123302 013737 177752 114150   MOV      HITMIS,RECDAT ;STORE REGISTER
20407 123310 052737 001000 177520   BIS      @1000,BCSR ;ENABLE HALT ON BREAK
20408 123316 032737 000004 114150   BIT      @BIT02,RECDAT ;LAST (R1) HIT?
20409 123324 001402          BEQ      5$         ;IF NOT, QBUS MEMORY
20410 123326 005202          INC      R2         ;INCREMENT 1K COUNT FOR PMI
20411 123330 000401          BR       6$         ;GO CONITNUE
20412 123332 005203 5$:      INC      R3         ;INCREMENT COUNT FOR QBUS MEM.
20413 123334 022701 154000 6$:      CMP      @154000,R1 ;LAST IN 4K PAGE?
20414 123340 101351          BHI      3$         ;IF NOT, BRANCH
20415 123342 022737 177600 172354   CMP      @177600,KIPAR6 ;2M BOUNDARY?
20416 123350 001337          BNE      1$         ;IF NOT, BRANCH
20417 123352 000402          BR       8$         ;IF 2M, DON'T TOUCH STACK
20418
20419          ;
20420          ; MAPPING IS DONE, FIND OUT HOW MANY PAGES WERE THERE
20421 123354 005726 7$:      TST      (SP)+    ;ADJUST STACK
20422 123356 005726          TST      (SP)+
20423 123360 010437 000004 8$:      MOV      R4,ERRVEC ;RESTORE ERROR VECTOR
20424 123364 005037 177572   CLR      MMRO        ;DISABLE MEMORY MANGEMENT
20425 123370 042737 000020 172516   BIC      @BIT04,MMR3 ;DISABLE 22 BITS
20426 123376 005702          TST      R2         ;ANY PMI MEMORY?
20427 123400 001405          BEQ      9$         ;IF NOT, GO CHECK QBUS MEMORY
20428 123402 006302          ASL      R2         ;TRANSLATE TO BYTES
20429 123404 010246          MOV      R2,-(SP)   ;STORE 1K # ON STACK
20430 123406 104405          TYPDS    ;TYPE # OF PAGES
20431 123410 104401 123434          TYPE    ,MEMK      ;TYPE ASCII
20432 123414 005703 9$:      TST      R3         ;ANY Q-BUS MEMORY?
20433 123416 001405          BEQ      10$        ;IF NOT, BRANCH
20434 123420 006303          ASL      R3         ;TRANSLATE TO BYTES
20435 123422 010346          MOV      R3,-(SP)   ;STORE 1K # ON STACK
20436 123424 104405          TYPDS    ;TYPE # OF PAGES
20437 123426 104401 123464          TYPE    ,MEMQ      ;TYPE ASCII
20438 123432 000431 10$:     BR       TST56      ;:EXIT TEST
20439
20440 123434 113 040 102 MEMK: .ASCIZ /K BYTES OF PMI MEMORY/<12><15>
      123437 131 124 105
      123442 123 040 117
      123445 106 040 120
      123450 115 111 040
      123453 115 105 115
      123456 117 122 131
    
```

TEST - MEMORY MAPPING

20441	123461	012	015	000
	123464	113	040	102
	123467	131	124	105
	123472	123	040	117
	123475	106	040	121
	123500	055	102	125
	123503	123	040	115
	123506	105	115	117
	123511	122	131	012
	123514	015	000	

MEMQ: .ASCIZ /K BYTES OF Q-BUS MEMORY/<12><15>

20442 .EVEN

20443
 20444 .SBTTL WRONG PARITY ABORT TEST
 20445 ;WRONG PARITY ABORT
 20446 ;THIS TEST VERIFIES ABORT TO 114 USING PARITY OR ECC MEMORY CSR.
 20447 ;IF MORE THEN 1 CSR PRESENT, ALL OF THEM WILL BE WRITTEN AT THE
 20448 ;SAME TIME.
 20449 ;

20450 ;ROUTINE TEST
 20451 ;. SIZE FOR ALL POSSIBLE MEMORY CSR'S
 20452 ;. STORE UP TO 16 CSR STARTING FROM TEMP
 20453 ;. WRITE ALL WITH WRONG PARITY
 20454 ;. WRITE TO 0
 20455 ;. READ IT BACK
 20456 ;. IF NO ABORT TO 114 THEN
 20457 ;. ERROR IN PARITY ABORT LOGIC
 20458 ;. ENDIF
 20459 ;. RESTORE CSR'S
 20460 ;ENDROUTINE

20461 ;*****
 20462 ;*****

20463 123516 000004 TST56: SCOPE
 20464 ;
 20465 ; FIND OUT ALL POSSIBLE MEMORY CSR LOCATIONS UP TO 16
 20466 ;

20466	123520	013701	000004		MOV	ERRVEC,R1	;STORE TIMEOUT VECTOR
20467	123524	012737	123562	000004	MOV	#2\$,ERRVEC	;POINT NEW TO PROGRAM
20468	123532	012737	000340	000006	MOV	#340,ERRVEC+2	;AT PRIORITY 7
20469	123540	005004			CLR	R4	;COUNT FOR CSR'S
20470	123542	012702	172100		MOV	#172100,R2	;FIRST POSSIBLE CSR
20471	123546	012703	002740		MOV	#TEMP,R3	;STORAGE LOCATION
20472	123552	005712			1\$: TST	(R2)	;IS CSR THERE?
20473	123554	010223			MOV	R2,(R3)+	;IF THERE, STORE
20474	123556	005204			INC	R4	;INCREMENT COUNT FOR CSR'S
20475	123560	000402			BR	3\$;BRANCH AROUND
20476	123562	005726			2\$: TST	(SP)+	;RESTORE STACK
20477	123564	005726			TST	(SP)+	
20478	123566	062702	000002		3\$: ADD	#2,R2	;POINT TO NEW CSR
20479	123572	022702	172136		CMP	#172136,R2	;ALL DONE?
20480	123576	101365			BHI	1\$;IF NOT, BRANCH
20481	123600	010137	000004		MOV	R1,ERRVEC	;RESTORE ERROR VECTOR
20482	123604	052737	001000	177746	BIS	#BIT09,CCR	;SET CACHE BYPASS

20483 ;
 20484 ; WRITE ALL CSR'S WITH WRONG ECC CODE
 20485 ; NOTE: IN PARITY MEMORY THOSE BITS ARE READ ONLY AND
 20486 ; DIAGNOSTIC MODE BIT FOR ECC IS THE SAME AS WRONG PARITY
 20487 ;

WRONG PARITY ABORT TEST

20488	123612	013701	000114		MOV	00114,R1	;STORE PARITY ABORT VECTOR	
20489	123616	012737	123704	000114	MOV	06\$,00114	;POINT NEW TO PROGRAM	
20490	123624	012737	000340	000116	MOV	0340,00116	;AT PRIORITY 7	
20491	123632	010402			MOV	R4,R2	;STORE CSR COUNT	
20492	123634	012703	002740		MOV	0TEMP,R3	;START WITH 1ST CSR	
20493	123640	052773	000005	000000	4\$:	BIS	0BIT02!BIT00,00(R3)	;DIAGNOSTIC OR WRONG PARITY
20494	123646	042733	003740		BIC	03740,0(R3)+	;CLEAR <10 5> CHECK BITS	
20495	123652	077406			SOB	R4,4\$;DO FOR ALL CSR'S	
20496	123654	005037	000000		CLR	000	;CLEAR TEST LOCATION WITH WRONG PR	
20497	123660	010204			MOV	R2,R4	;RESTORE COUNTER	
20498	123662	012703	002740		MOV	0TEMP,R3	;START WITH 1ST CSR	
20499	123666	042733	000004		5\$:	BIC	0BIT02,0(R3)+	;CLEAR ALL WRONG PARITY
20500	123672	077203			SOB	R2,5\$;IN ALL CSR'S	
20501	123674	005737	000000		TST	000	;READ BACK WRONG PARITY	
20502	123700	104034			ERROR	+34	;NO WRONG PARITY ABORT	
20503	123702	000402			BR	7\$		
20504	123704	005726			6\$:	TST	(SP)+	;ADJUST STACK
20505	123706	005726			TST	(SP)+		
20506	123710	012703	002740		7\$:	MOV	0TEMP,R3	;START WITH 1ST CSR
20507	123714	042733	000001		8\$:	BIC	0BIT00,0(R3)+	;CLEAR ALL WRONG PARITY
20508	123720	077403			SOB	R4,8\$;IN ALL CSR'S	
20509	123722	032737	100000	177744		BIT	0BIT15,MSER	;ABORT REFLECTED IN MSER?
20510	123730	001004			BNE	9\$;IF YES, BRANCH	
20511	123732	012737	100000	001124		MOV	0100000,\$GDDAT	
20512	123740	104035			ERROR	+35	;MSER<15> NOT SET ON PARITY ABORT	
20513	123742	042737	001000	177746	9\$:	BIC	0BIT09,CCR	;CLEAR CACHE BYPASS
20514	123750	005037	000000		CLR	000	;CLEAR WRONG PARITY	
20515	123754	005037	177744		CLR	MSER	;CLEAR MSER	
20516	123760	010137	000114		MOV	R1,00114	;RESTORE PARITY ABORT	
20517								
20518								

TEST - DMA TAG PARITY IN STANDALONE MODE

```

20520 .SBTTL TEST DMA TAG PARITY IN STANDALONE MODE
20521 ;CHECK DMA TAG STORE PARITY BIT.
20522 ;ROUTINE TEST
20523 ;. CACHE DMA_PARITY
20524 ;. LET BCSR<08>=#1
20525 ;. REPORT ALL ERRORS
20526 ;ENDROUTINE
20527 ;
20528 ;ROUTINE DMA PARITY
20529 ;. GENERATE PARITY ERRORS
20530 ;. CHECK MSER<13>
20531 ;. LET BCSR<08>=#0
20532 ;ENDROUTINE
20533
20534 ;*****
123764 000004
20535 123766 000240
20536 123770 005737 003032
20537 123774 001002
20538 123776 000240
20539 124000 000456
20540 124002 000240
20541
20542 ;
20543 ; ALLOCATE CODE IN CACHE
20544 124004 012702 124040
20545 124010 042737 001000 177520
20546 124016 005722
20547 124020 022702 124100
20548 124024 001374
20549 124026 005737 002740
20550 124032 013737 177520 002730
20551
20552 ;
20553 ; IN STANDALONE MODE TRY TO VERIFY RESPONSE TO PARITY ERRORS
20554 124040 052737 000400 177520
20555 124046 052737 002001 177746
20556 124054 005037 002740
20557 124060 013705 177744
20558 124064 042737 002000 177746
20559 124072 042737 000400 177520
20560
20561 ;
20562 ; RETURN FROM STANDALONE MODE
20563 124100
20564 124100 022705 060020
20565 124104 001401
20566 124106 104117
20567 124110 005037 177744
20568 124114 012737 000400 177746
20569 124122 013737 002730 177520
20570 124130 052737 001000 177520
20571

```

TST57: SCOPE
NOP
TST CCHPAS ;have done enough inclusive passes?
BNE 99\$; not yet
NOP ; debug aid
BR TST60 ;;GO TO NEXT TEST
99\$: NOP
; MOV #DMAPAR,R2 ;POINT TO STANDALONE CODE
; BIC #1000,BCSR ;DISABLE HALT ON BREAK
1\$: TST (R2)+ ;ALLOCATE IN CACHE
CMP #DPAREN,R2 ;LAST ADDRESS?
BNE 1\$;IF NOT, BRANCH
TST TEMP ;ALLOCATE TEST LOCATION
MOV BCSR,SAVBR ;SAVE BCSR
; DMAPAR: BIS #BIT08,BCSR ;SET STANDALONE MODE BIT
; BIS #BIT10:BIT00,CCR ;WRITE WRONG TAG PARITY
; CLR TEMP ;WRITE HIT WITH WRONG PARITY
; MOV MSER,R5 ;STORE MSER
; BIC #BIT10,CCR ;CLEAR WRONG PARITY BIT
2\$: BIC #BIT08,BCSR ;CLEAR STANDALONE BIT
; DPAREN: CMP #60020,R5 ;WRONG DMA PARITY?
; BEQ 4\$;IF OK, BRANCH
; ERROR +117 ;MSER NOT SET IN STANDALONE MODE
4\$: CLR MSER
; MOV #400,CCR ;FLUSH THE CACHE
; MOV SAVBR,BCSR ;RESTORE BCSR
; BIS #1000,BCSR ;ENABLE HALT ON BREAK

TEST - DMA TAG PARITY W/O STANDALONE MODE

```

20573 .SBTTL TEST DMA TAG PARITY W/O STANDALONE MODE
20574 ;CHECK DMA TAG PARITY BIT WITHOUT GOING INTO STANDALONE MODE.
20575 ;
20576 ;CCR <10> WRITE WRONG TAG PARITY
20577 ;
20578 ;ROUTINE TEST
20579 ;. LET CCR<10>=#1
20580 ;. ALLOCATE LOCATION IN CACHE
20581 ;. INITIATE DMA WRITE TRANSFERS
20582 ;. IF MSER<04> NE #1 THEN
20583 ;. . ERROR
20584 ;. ENDF
20585 ;ENDROUTINE
20586
20587 ;*****
20588 124136 000004 TST60: SCOPE
20589 124140 000240 NOP
20590 124142 005737 003032 TST CCHPAS ;have done enough inclusive passes?
20591 124146 001002 BNE 99$ ; not yet
20592 124150 000240 NOP ; debug aid
20593 124152 000471 BR TST61 ;;GO TO NEXT TEST
20594 124154 000240 99$: NOP
20595 124156 032737 000200 000052 BIT #BIT07,#52 ;UFD MODE?
20596 124164 001402 BEQ 110$ ;IF NOT, BRANCH
20597 124166 000137 140340 JMP $EOP ;OTHERWISE, NEXT PASS
20598 124172 005737 002664 110$: TST CSR1 ;AT LEAST ONE Q22BE FOUND?
20599 124176 001457 BEQ TST61 ;;IF NOT, EXIT TEST
20600 ;
20601 ; ALLOCATE TEST IN CACHE
20602 ;
20603 124200 012703 124200 11$: MOV #.,R3 ;START WITH CURRENT INSTRUCTION
20604 124204 005723 10$: TST (R3)+ ;READ A WORD
20605 124206 022703 124316 CMP #4$,R3 ;ALL DONE?
20606 124212 001374 BNE 10$ ;IF NOT, CONTINUE
20607 ;
20608 ; WRITE A WORD WITH WRONG PARITY
20609 ;
20610 124214 013737 000114 001160 1$: MOV #0114,$TMP0 ;STORE PARITY VECTOR
20611 124222 012737 124270 000114 MOV #2$,#0114 ;POINT TO TEST AREA
20612 124230 052737 002000 177746 BIS #BIT10,CCR ;WRITE WRONG TAG PARITY
20613 124236 005037 002740 CLR TEMP ;WRITE MISS WITH WRONG TAG PARITY
20614 124242 042737 002000 177746 BIC #BIT10,CCR ;CLEAR WRONG TAG PARITY
20615 124250 052737 000200 177746 BIS #BIT07,CCR ;PARITY ABORT
20616 124256 005000 CLR R0 ;FLAG TO DO 1 TRANSFER
20617 124260 004737 137370 JSR PC,DMATRN ;DO DMA WRITE TO TEMP THRU Q22BE
20618 124264 104116 ERROR +116 ;NO PARITY ABORT
20619 124266 000413 BR 4$ ;BRANCH TO TEST MSER
20620 124270 013704 177744 2$: MOV MSER,R4 ;STORE REGISTER
20621 124274 005037 177744 CLR MSER ;CLEAR MSER
20622 124300 005726 TST (SP)+ ;
20623 124302 005726 TST (SP)+ ;RESTORE STACK
20624 124304 005726 TST (SP)+ ;
20625 124306 032704 000020 3$: BIT #BIT04,R4 ;MSER OK?
20626 124312 001001 BNE 4$ ;IF SET, BRANCH
20627 124314 104116 ERROR +116 ;MSER<4> NOT SET
20628 124316 005037 177744 4$: CLR MSER ;CLEAR MSER

```

TEST - DMA TAG PARITY W/O STANDALONE MODE

20629 124322 012737 000400 177746
20630 124330 013737 001160 000114

MOV @400,CCR
MOV \$TMP0,@#114

;FLUSH CACHE
;RESTORE VECTOR

TEST - DMA WRITE HIT CYCLES

```

20632 .SBTTL TEST - DMA WRITE HIT CYCLES
20633 ;CHECK THAT DMA WRITE HITS INVALIDATE CACHE.
20634 ;ROUTINE TEST
20635 ;. ALLOCATE A LOCATION IN CACHE
20636 ;. INITIATE DMA WRITE TO THIS LOCATION
20637 ;. READ THIS LOCATION BACK
20638 ;. IF IT IS CHANGED OR HIT/MISS EQ HIT
20639 ;. ERROR
20640 ;.
20641 ;. ENDIF
20642 ;. WRITE TO THE FIRST 16 LOCATION THEIR ADDRESS
20643 ;. DO BLOCK MODE TRANSFER TO THOSE LOCATIONS
20644 ;. START READ WITH THE LAST LOCATION
20645 ;. IF RECORD ANY HITS THEN
20646 ;. ERROR
20647 ;.
20648 ;. ENDIF
20649 ;. INITIATE READ DMA TO THE SAME TEST LOCATIONS
20650 ;. READ THEM BACK
20651 ;. IF HIT/MISS NE HIT THEN
20652 ;. ERROR
20653 ;.
20654 ;. ENDIF
;ENDROUTINE

;*****
TST61: SCOPE
      NOP
20655 124336 000004
20656 124340 000240
20657 124342 005737 003032
20658 124346 001003
20659 124350 000240
20660 124352 000137 125304
20661 124356 000240
20662 124360 032737 000200 000052
20663 124366 001402
20664 124370 000137 140340
20665 124374 005737 002664
20666 124400 001002
20667 124402 000137 140340
20668
20669 ; TRY TO DO DMA WRITE AT ALL DIFFERENT PRIORITIES
20670 ;
20671 124406 012702 000340
20672 124412 000402
20673 124414 162702 000040
20674 124420 005037 002740
20675 124424 005000
20676 124426 106402
20677 124430 004737 137370
20678 124434 042737 001000 177520
20679 124442 005737 002740
20680 124446 013737 177752 114150
20681 124454 052737 001000 177520
20682 124462 032737 000004 114150
20683 124470 001401
20684 124472 104120
20685 124474 022737 012525 002740
20686 124502 001401
20687 124504 104123
      BIT      #BIT07,#052          ;UFD MODE?
      BEQ      1#                    ;IF NOT, BRANCH
      JMP      #EOP                  ;OTHERWISE, NEXT PASS
2# :   TST      CSR1                 ;AT LEAST 1 Q22BE?
      BNE      2#                    ;IF YES, BRANCH
      JMP      #EOP                  ;OTHERWISE, NEXT PASS
99# :   NOP
      CCHPAS          ;have done enough inclusive passes?
      99#             ; not yet
      ALLEND          ; debug aid
      99#             ; yes skip this
      MOV      #340,R2              ;START WITH 7
      BR      4#                    ;GO DO IT
      SUB      #40,R2               ;LOWER PRIORITY
      CLR      TEMP                 ;CLEAR TEST LOCATION
      CL      R0                    ;DO JUST 1 WORD
      MTPS    R2                   ;LOWER PRIORITY
      JSR     PC,DMATRN             ;DO DATO
      BIC     #1000,BCSR            ;DISABLE HALT ON BREAK
      TST     TEMP                 ;STILL CACHED?
      MOV     HITMIS,RECDAT        ;STORE HIT/MISS
      BIS     #1000,BCSR            ;ENABLE HALT ON BREAK
      BIT     #BIT02,RECDAT        ;LAST ACCESS HIT?
      BEQ     5#                    ;IF MISS, BRANCH
      ERROR   +120
      CMP     #12525,TEMP           ;DATO OK?
      BEQ     6#                    ;IF SO, BRANCH
      ERROR   +127                 ;DATO

```

TEST - DMA WRITE HIT CYCLES

```

20688 124506 005702          6#:   TST      R2          ;LAST PRIORITY 0?
20689 124510 001341          ;BNE      3#          ;IF NOT, BRANC.
20690 124512 106427 000340    ;MTPS     #340       ;RESTORE PRIORITY
20691          ;
20692          ;
20693          ; VERIFY THAT BYPASS WITH DMA DATI INVALIDATES CACHE
20694          ;
20695 124516 012737 001000 177746 8#:   MOV      #BIT09,CCR ;SET BYPASS
20696 124524 004737 137370    ;JSR     PC,DMATRN   ;DO DMA DATI
20697 124530 005037 177746    ;CLR     CCR         ;CLEAR BYPASS
20698 124534 042737 001000 177520 ;BIC     #1000,BCSR  ;DISABLE HALT ON BREAK
20699 124542 005737 002740    ;TST     TEMP        ;IN CACHE?
20700 124546 013737 177752 114150 ;MOV     HITMIS,RECDAT ;STORE REGISTER
20701 124554 052737 001000 177520 ;BIS     #1000,BCSR  ;ENABLE HALT ON BREAK
20702 124562 032737 000004 114150 ;BIT     #BIT02,RECDAT ;TEMP WAS A HIT?
20703 124570 001401          ;BEQ     DATBO       ;IF NOT, BRANCH
20704 124572 104123          ;ERROR   +123
20705          ;
20706          ; DO DATBO
20707          ;
20708 124574 012701 000010    DATBO: MOV     #10,R1   ;COUNTER FOR 8
20709 124600 012702 002740    ;MOV     #TEMP,R2   ;START WITH TEMP
20710 124604 005022          1#:   CLR     (R2)+    ;CLEAR ALL 16
20711 124606 077102          ;SOB     R1,1#      ;DO ALL 16
20712 124610 005200          ;INC     R0         ;FLAG BLOCK MODE
20713 124612 012777 002740 056050 ;MOV     #TEMP,@BA  ;LOAD DMA ADDRESS
20714 124620 012777 177770 056044 ;MOV     #177770,@WC ;DO 8 WORDS
20715 124626 012777 001701 056030 ;MOV     #1701,@CSR1 ;16 WORDS FROM 32K
20716 124634 004737 137370    ;JSR     PC,DMATRN  ;DO DATBO
20717 124640 032777 010000 056020 ;BIT     #BIT12,@CSR2 ;NO BLOCK MODE SLAVE?
20718 124646 001401          ;BEQ     2#         ;IF NOT, BRANCH
20719 124650 104123          ;ERROR   +123      ;NO BLOCK MODE SLAVE
20720 124652 012701 000004    2#:   MOV     #4,R1    ;COUNTER FOR 4 LOCATIONS
20721 124656 012702 002740    ;MOV     #TEMP,R2   ;START WITH TEMP
20722 124662 042737 001000 177520 3#:   BIC     #1000,BCSR ;DISABLE HALT ON BREAK
20723 124670 005712          ;TST     (R2)       ;ACCESS A LOCATION
20724 124672 013737 177752 114150 ;MOV     HITMIS,RECDAT ;STORE REGISTER
20725 124700 052737 001000 177520 ;BIS     #1000,BCSR  ;ENABLE HALT ON BREAK
20726 124706 032737 000004 114150 ;BIT     #BIT02,RECDAT ;HIT?
20727 124714 001401          ;BEQ     4#         ;IF NOT, BRANCH
20728 124716 104121          ;ERROR   +121      ;DMA DOES NOT INVALIDATE
20729 124720 022722 012525    4#:   CMP     #12525,(R2)+ ;DATO OK?
20730 124724 001401          ;BEQ     5#         ;IF SO, BRANCH
20731 124726 104123          ;ERROR   +123      ;IN DMA
20732 124730 062702 000002    5#:   ADD     #2,R2    ;DO IN 2 WORDS
20733 124734 077126          ;SOB     R1,3#     ;DO ALL 16
20734          ;
20735          ;
20736          ; DO 4K OF DATO AND CHECK FOR MISS
20737          ;
20738 124736 004737 136574          ;JSR     PC,INITMM  ;SET UP MMU
20739 124742 012737 002000 172354 ;MOV     #2000,KIPAR6 ;START AT 32K
20740 124750 042737 100000 172314 ;BIC     #BIT15,KIPDR6 ;NO BYPASS
20741 124756 052737 000001 177572 ;BIS     #BIT00,MMRO ;ENABLE MMU
20742 124764 012737 125124 000004 ;MOV     #DATI,@#4  ;IF 32K NXW
20743 124772 012702 140000    ;MOV     #140000,R2 ;START WITH 32K
20744 124776 005712          ;TST     (R2)      ;EXIT

```

TEST DMA WRITE HIT CYCLES

20745	125000	012701	010000		MOV	#10000,R1		:COUNTER FOR 4K
20746	125004	005022			CLR	(R2)+		:CLEAR ALL 16
20747	125006	077102		6#:	SOB	R1,6#		:DO ALL 16
20748	125010	005200			INC	R0		:FLAG BLOCK MODE
20749	125012	012777	000000	055650	MOV	#0,#BA		:LOAD DMA ADDRESS
20750	125020	012777	170000	055644	MOV	#-10000,#WC		:DO 4K
20751	125026	012777	003701	055630	MOV	#3701,#CSR1		:16 WORDS FROM 32K
20752	125034	004737	137370		JSR	PC,DMATR		:DO DATBO
20753	125040	012701	004000		MOV	#4000,R1	7#:	:COUNTER FOR 4K LOCATIONS
20754	125044	012702	140000		MOV	#140000,R2		:START WITH 0
20755	125050	042737	001000	177520	BIC	#1000,BCSR	8#:	:DISABLE HALT ON BREAK
20756	125056	005712			TST	(R2)		:ACCESS A LOCATION
20757	125060	013737	177752	114150	MOV	HITMIS,RECDAT		:STORE REGISTER
20758	125066	052737	001000	177520	BIS	#1000,BCSR		:ENABLE HALT ON BREAK
20759	125074	032737	000004	114150	BIT	#BIT02,RECDAT		:HIT?
20760	125102	001401			BEQ	9#		:IF NOT, BRANCH
20761	125104	104121			ERROR	+121		:DMA DOES NOT INVALIDATE
20762	125106	022722	012525		CMP	#12525,(R2)+	9#:	:DATO OK?
20763	125112	001401			DEQ	10#		:IF SO,BRANCH
20764	125114	104123			ERROR	+123		:IN DMA
20765	125116	062702	000002		ADD	#2,R2	10#:	:DO IN 2 WORDS
20766	125122	077126			SOB	R1,8#		:DO ALL OF THEM
20767					:			
20768					: CHECK DATI			
20769					:			
20770	125124	005037	177572		DATI:	CLR	MMR0	:DISABLE MMU
20771	125130	012706	001100		MOV	#1100,SP		:RESTORE STACK
20772	125134	012737	137542	000004	MOV	#TOUT,##4		:AND TIMEOUT
20773	125142	012737	052525	002740	MOV	#52525,TEMP		:LOAD MEMORY
20774	125150	005000			CLR	R0		:JUST 1 WORD
20775	125152	004737	137452		JSR	PC,DMARD		:DO DATI
20776	125156	022777	052525	055510	CMP	#52525,#DATA		:DATI OK?
20777	125164	001401			BEQ	11#		:IF YES, BRANCH
20778	125166	104123			ERROR	+123		:DATI
20779					:			
20780					: DO DATBI			
20781					:			
20782	125170	012701	000010		11#:	MOV	#10,R1	:COUNTER FOR 16 LOCATIONS
20783	125174	012702	002740		MOV	#TEMP,R2		:START WITH TEMP
20784	125200	012703	002740		MOV	#TEMP,R3		:START WITH TEMP
20785	125204	010322			12#:	MOV	R3,(R2)+	:PUT ADDRESSES
20786	125206	005723			TST	(R3)+		:INCREMENT ADDRESS
20787	125210	077103			SOB	R1,12#		:DO ALL 16
20788	125212	005200			INC	R0		:FLAG BLOCK MODE
20789	125214	004737	137452		JSR	PC,DMARD		:DO DATBI
20790	125220	032777	010000	055440	BIT	#BIT12,#CSR2		:NO BLOCK MODE SLAVE?
20791	125226	001401			BEQ	13#		:IF NO, BRANCH
20792	125230	104123			ERROR	+123		
20793	125232	022777	002756	055434	13#:	CMP	#TEMP+16,#DATA	:DATI OK?
20794	125240	001401			BEQ	14#		:IF SO, BRANCH
20795	125242	104123			ERROR	+123		:IN DATIB
20796	125244	042737	001000	177520	14#:	BIC	#1000,BCSR	:DISABLE HALT ON BREAK
20797	125252	005737	002740		TST	TEMP		:ACCESS
20798	125256	013737	177752	114150	MOV	HITMIS,RECDAT		:STORE REGISTER
20799	125264	052737	001000	177520	BIS	#1000,BCSR		:ENABLE HALT ON BREAK
20800	125272	032737	000004	114150	BIT	#BIT02,RECDAT		:HIT?
20801	125300	001001			BNE	15#		:IF YES, BRANCH

TEST DMA WRITE HIT CYCLES

20802 125302 104123
20803 125304
20804 125304

ERROR +123
151:
ALLEND:

TEST - DIFFERENT LEVELS OF INTERRUPTS

```

20806 .SBTTL TEST - DIFFERENT LEVELS OF INTERRUPTS
20807 ;DIFFERENT LEVELS OF INTERRUPTS
20808 ;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
20809 ;LEVELS. ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
20810 ;PIRQ'S WILL BE TESTED.
20811 ;
20812 ;CHECK DIFFERENT LEVELS OF INTERRUPTS.
20813 ;ROUTINE TEST
20814 ;. SET VECTOR TO INTERRUPT_DMA
20815 ;. FOR INTERRUPTS FROM 4 TO 7 DO
20816 ;. . ENABLE INTERRUPTS
20817 ;. . SET PRIORITY=INTERUPT
20818 ;. . IF INTERRUPT_FLAG SET THEN
20819 ;. . . ERROR
20820 ;. . . ENDF
20821 ;. . . ENABLE INTERRUPTS
20822 ;. . . SET PRIORITY=INTERRUPT 1
20823 ;. . . IF INTERRUPT_FLAG NOTSET THEN
20824 ;. . . . ERROR
20825 ;. . . . ENDF
20826 ;. . . . LET INTERRUPT_DMA=0
20827 ;. . . . ENDDO
20828 ;ENDROUTINE
20829 ;
20830 ;ROUTINE INTERUPT_DMA
20831 ;. LET INTERRUPT_FLAG=1
20832 ;RETURN
20833 ;ENDROUTINE
20834
20835 ;*****
20836 125304 000004 TST62: SCOPE BIT #BIT07,#52 ;UFD MODE?
20837 125306 032737 000200 000052 BNE TST63 ;:IF SO, EXIT TEST
20838 125314 001122 TST CSR1 ;AT LEAST ONE Q22BE FOUND?
20839 125316 005737 002664 BEQ TST63 ;:IF NOT, EXIT TEST
20840
20841 ;
20842 ; SETUP INITIAL PRIORITY TO 7
20843 ;
20843 125324 013703 002664 MOV CSR1,R3 ;DO FOR FIRST FOUND Q22BE
20844 125330 012777 125424 055340 MOV #5,$VQBE1 ;POINT INTERRUPT VECTOR TO PROGRAM
20845 125336 012777 000340 055334 MOV #340,$VQPR1 ;AT PRIORITY 7
20846 125344 012700 003002 MOV #Q22EN,R0 ;START WITH 7 FOR INTERRUPTS
20847 125350 012701 000340 MOV #340,R1 ;LOW BOUNDARY FOR NO INTERRUPTS
20848 125354 000402 BR 2$ ;TRY TO DO IT FOR FIRST
20849 125356 162701 000040 1$: SUB #40,R1 ;LOWER LOW BOUNDARY
20850 ;
20851 ; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
20852 ;
20853 125362 012737 000340 001160 2$: MOV #340,$TMP0 ;TOP PRIORITY FOR NO INTERRUPTS
20854 125370 000403 BR 4$ ;DO FIRST ONE
20855 125372 162737 000040 001160 3$: SUB #40,$TMP0 ;DO AT NEXT LEVEL
20856 125400 106437 001160 4$: MTPS $TMP0 ;SET PRIORITY NOT TO INTERRUPT
20857 125404 004737 137350 JSR PC,Q22INT ;ENABLE INTERRUPTS
20858 125410 012077 055252 MOV (R0)+,$CSR2 ;CLEAR GO BIT
20859 125414 000240 NOP
20860 125416 000240 NOP
20861 125420 000240 NOP

```

TEST DIFFERENT LEVELS OF INTERRUPTS

```

20862 125422 000403          BR      6$
20863 125424 104126          5$:  ERROR +126          ;IF NO INTERUPT, BRANCH
20864 125426 005726          TST    (SP)+          ;INTERRUPTS HAPPEN
20865 125430 005726          TST    (SP)+          ;RESTORE STACK
20866 125432 020137 001160    6$:  CMP    R1,$TMPO          ;LAST ONE?
20867 125436 001355          BNE    3$              ;IF NOT BRANCH
20868 125440 022710 000002    CMP    #2,(R0)         ;AT BR4?
20869 125444 001344          BNE    1$              ;IF NOT LAST ONE, BRANCH
20870
20871          ;
20872          ; INTERRUPT AT ALL LEVELS
20873 125446 012777 125542 055222 INQ22: MOV    #5,$@VQBE1          ;POINT INTERRUPT VECTOR TO PROGRAM
20874 125454 012777 000340 055216    MOV    #340,$@VQPR1      ;AT PRIORITY 7
20875 125462 012700 003002          MOV    #Q22EN,R0         ;START WITH 7 FOR INTERRUPTS
20876 125466 012701 000300          MOV    #300,R1           ;TOP BOUNDARY FOR INTERRUPTS
20877 125472 000402          BR     2$              ;TRY TO DO IT FOR FIRST
20878 125474 162701 000040    1$:  SUB    #40,R1         ;LOWER TOP BOUNDARY
20879
20880          ;
20881          ; CHECK THAT INTERRUPTS HAPPEN AT PRIORITY LOWER THAN BR
20882 125500 010137 001160    2$:  MOV    R1,$TMPO          ;PRIORITY FOR INTERRUPTS
20883 125504 000403          BR     4$              ;DO FIRST ONE
20884 125506 162737 000040 001160  3$:  SUB    #40,$TMPO          ;DO AT NEXT LEVEL
20885 125514 106437 001160    4$:  MTPS  $TMPO           ;SET PRIORITY NOT TO INTERRUPT
20886 125520 004737 137350          JSR    PC,Q22INT         ;ENABLE INTERRUPTS
20887 125524 011077 055136          MOV    (R0),@CSR2       ;CLEAR GO BIT
20888 125530 000240          NOP
20889 125532 000240          NOP
20890 125534 000240          NOP
20891 125536 104126          ERROR +126            ;INTERRUPTS DON'T HAPPEN
20892 125540 000402          BR     6$              ;DON T RESTORE STACK
20893 125542 005726          5$:  TST    (SP)+          ;RESTORE STACK
20894 125544 005726          TST    (SP)+
20895 125546 005737 001160    6$:  TST    $TMPO           ;LAST ONE 0?
20896 125552 001355          BNE    3$              ;IF NOT BRANCH
20897 125554 022720 000002    CMP    #2,(R0)+        ;AT BR4?
20898 125560 001345          BNE    1$              ;IF NOT LAST ONE, BRANCH
20899
20900

```

TEST - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

```

20902 .SBTTL TEST - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
20903 ;CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.
20904 ;ROUTINE TEST
20905 ; IF UFD THEN
20906 ;. EXIT TEST
20907 ;. ENDF
20908 ;. DO FOR I FROM #6 DOWN TO #3
20909 ;. SET PRIORITY TO I
20910 ;. ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
20911 ;. IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
20912 ;. ERROR
20913 ;. ENDF
20914 ;. ENDDO
20915 ;ENDROUTINE
20916
20917

```

```

20918 125562 000004 TST63: SCOPE
20919 125564 032737 000200 000052 BIT #BIT07,#52 ;UFD MODE?
20920 125572 001065 BNE TST64 ;;IF SO, EXIT TEST
20921 125574 005737 002664 TST CSR1 ;AT LEAST ONE Q22BE FOUND?
20922 125600 001462 BEQ TST64 ;;IF NOT, EXIT TEST
20923 125602 012777 125710 055066 MOV #3#,#VQBE1 ;SETUP Q22BE VECTOR
20924 125610 012777 000340 055062 MOV #340,#VQPR1 ;AT PRIORITY 7
20925 125616 012737 125720 000240 MOV #4#,#PIRQVEC ;SETUP PIRQ VECTOR
20926 125624 012737 000340 000242 MOV #340,#PIRQVEC+2 ;AT PRIORITY 7
20927 125632 012700 003002 MOV #Q22EN,R0 ;POINT THRU PRIORITIES FOR Q22BE
20928 125636 012704 125736 MOV #PIRQT,R4 ;POINTER THRU PIRQ'S
20929 125642 013703 002664 MOV CSR1,R3 ;DO FOR FIRST Q22BE
20930 125646 012702 000300 MOV #300,R2 ;START WITH CPU PRIORITY AT 7
20931 125652 000402 BR 2# ;DO FIRST ONE
20932 125654 162702 000040 1#: SUB #40,R2 ;LOWER CPU PRIORITY
20933 125660 106427 000340 2#: MTPS #340 ;RAISE PRIORITY TO 7
20934 125664 012437 177772 MOV (R4)+,#PIRQ ;SET PRIORITY FOR PIRQ'S
20935 125670 004737 137350 JSR PC,Q22INT ;INITIALISE Q22BE TO INTERRUPT
20936 125674 012077 054766 MOV (R0)+,#CSR2 ;SET DONE BIT
20937 125700 106402 MTPS R2 ;LOWER PRIORITY
20938 125702 000240 NOP
20939 125704 000240 NOP
20940 125706 000240 NOP
20941 125710 104124 3#: ERROR +124 ;PIRQ'S DON'T TAKE OVER BIRQ S
20942 125712 005726 TST (SP)+ ;CLEAN UP STACK
20943 125714 005726 TST (SP)+
20944 125716 000402 BR 5# ;BRANCH AROUND PIRQ INTERRUPT
20945 125720 005726 4#: TST (SP)+ ;CLEAN UP STACK
20946 125722 005726 TST (SP)+
20947 125724 022702 000140 5#: CMP #140,R2 ;PRIORITY 3 LAST ONE?
20948 125730 001351 BNE 1# ;IF NOT BRANCH
20949 125732 005037 177772 CLR PIRQ ;CLEAR ANY REQUESTS
20950 125736 100000 040000 020000 PIRQT: .WORD 100000,40000,20000,10000 ;PIRQ'S<7-4>
20951 125744 010000

```

TEST POWER DOWN TEST

```

20953 .SBTTL TEST - POWER DOWN TEST
20954 ;USING Q22BE THIS TEST WILL CHECK THAT ON POWER DOWN CONDITION IF
20955 ;POWER UP CODE 00 IS SELECTED THE CPU TRAPS THRU 24
20956 .ROUTINE TEST
20957 ; IF UFD OR POWER UP CODE 00 NOT SELECTED THEN
20958 ;. EXIT TEST
20959 ;. ENDF
20960 ;. SET 24 TO POINT TO TEST AREA
20961 ;. LET CSR2<5> = #1 TO NEGATE BPOK
20962 ;. IF NO TRAP TO 24 THEN
20963 ;. ERROR IN POWER DOWN CYCLE
20964 ;. ENDF
20965 ;. IF TRAP TO 24 THEN
20966 ;. LET CSR2<5> = #0
20967 ;. ENDF
20968 ;ENDROUTINE
20969
20970 ;*****
125746 000004
20971 125750 032737 000200 000052 TST64: SCOPE
20972 125756 001033 BIT #BIT07,#52 ;UFD MODE?
20973 125760 005737 002664 BNE TST65 ;;EXIT TEST IN UFD MODE
20974 125764 001430 TST CSR1 ;AT LEAST ONE Q22BE FOUND?
20975 125766 013737 000024 001160 BEQ TST65 ;;IF NOT, EXIT TEST
20976 125774 012737 126030 000024 MOV PWRVEC,#TMP0 ;SAVE POWER UP VECTOR
20977 126002 012737 000340 000026 MOV #1#,PWRVEC ;POINT NEW TO PROGRAM
20978 126010 012777 000040 054650 MOV #340,PWRVEC+2 ;AT PRIORITY 7
20979 126016 000240 NOP ;DO POWER DOWN
20980 126020 005077 054642 CLR @CSR2 ;CLEAR POWER DOWN BIT
20981 126024 104125 ERROR +125 ;NO POWER DOWN TRAP
20982 126026 000404 BR 2# ;SKIP RESTORING STACK
20983 126030 005077 054632 1#: CLR @CSR2 ;CLEAR POWER DOWN BIT
20984 126034 005726 TST (SP)+ ;RESTORE STACK POINTER
20985 126036 005726 TST (SP)+
20986 126040 013737 001160 000024 2#: MOV #TMP0,PWRVEC ;RESTORE POWER VECTOR
20987
20988
20989

```


TEST - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS

```

20991 .SBTTL TEST - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS
20992 ;IF TWO Q22BE ARE AVAILABLE, THIS TEST WILL CHECK THAT HIGHER LEVEL
20993 ;INTERRUPT REQUESTS TAKE PRIORITY OVER LOWER LEVEL ONES
20994 ;ROUTINE TEST
20995 ;. IF UFD OR 2ND Q22BE NOT AVAILABLE THEN
20996 ;. EXIT TEST
20997 ;. ENDF
20998 ;. DO FOR PRIORITY_LEVELS FROM 4 TO 6
20999 ;. SET UP 1ST Q22BE TO INTERRUPT AT PRIORITY_LEVEL
21000 ;. SET UP 2ND Q22BE TO INTERRUPT AT PRIORITY_LEVEL+1
21001 ;. SET UP CPU PRIORITY TO PRIORITY_LEVEL-1
21002 ;. IF INTERRUPTS FORM 1ST Q22BE HAPPENED BEFORE 1ST THEN
21003 ;. ERROR
21004 ;. ENDF
21005 ;. ENDDO
21006 ;ENDROUTINE
21007
21008 ;*****
126046 000004
21009 126050 032737 000200 000052 TST65: SCOPE
21010 126056 001064 BIT #BIT07,#52 ;UFD MODE?
21011 126060 005737 002704 BNE TST66 ;;IF SO, EXIT TEST
21012 126064 001002 TST CSR12 ;SECOND Q22BE AVAILABLE?
21013 126066 000137 140340 BNE 1$ ;IF YES, GO DO TEST
21014 ; JMP $EOP ;OTHERWISE, GOTO EOP
21015 ; INITIALISE VECTORS FOR Q22BE'S
21016 ;
21017 126072 012777 126206 054576 1$: MOV #4,$,BVQBE1 ;VECTOR FOR 1ST ONE
21018 126100 012777 000340 054572 MOV #340,$,BVQPR1 ;AT PRIORITY 7
21019 126106 012777 126216 054602 MOV #5,$,BVQBE2 ;VECTOR FOR 2ND ONE
21020 126114 012777 000340 054576 MOV #340,$,BVQPR2 ;AT PRIORITY 7
21021 126122 012700 003004 MOV #Q22EN+2,R0 ;POINTER FOR Q22BE BR'S
21022 126126 012702 000240 MOV #240,R2 ;CPU AT 5
21023 126132 000402 BR 3$ ;START DOING ARBITRATION
21024 ;
21025 ; DO FOR CPU PRIORITIES 5-3
21026 ;
21027 126134 162702 000040 2$: SUB #40,R2 ;LOWER CPU PRIORITY
21028 126140 106427 000340 3$: MTPS #340 ;CPU AT 7
21029 126144 013703 002704 MOV CSR12,R3 ;Q22BE 2 AT HIGHER BR
21030 126150 004737 137350 JSR PC,Q22INT ;INITIALISE TO INTERRUPTS
21031 126154 011077 054506 MOV (R0),@CSR2 ;START 1ST ONE
21032 126160 013703 002664 MOV CSR1,R3 ;Q22BE 1 A; LOWER BR
21033 126164 005740 TST -(R0) ;HIGHER PRIORITY
21034 126166 004737 137350 JSR PC,Q22INT ;INITIALISE
21035 126172 012077 054510 MOV (R0)+,@CSR22 ;START 2ND ONE
21036 126176 106402 MTPS R2 ;LOWER CPU PRIORITY
21037 126200 000240 NOP ;WAIT A WHILE
21038 126202 000240 NOP
21039 126204 000240 NOP
21040 126206 104126 4$: ERROR +126 ;INTERRUPTS IN WRONG ORDER
21041 126210 005726 TST (SP)+ ;RESTORE STACK
21042 126212 005726 TST (SP)+
21043 126214 000402 BR 6$
21044 126216 005726 5$: TST (SP)+ ;RESTORE STACK
21045 126220 005726 TST (SP)+
21046 126222 022702 000140 6$: CMP #140,R2 ;PRIORITY 3 LAST?
    
```

TEST - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS

21047 126226 001342
21048

BNE 24

;IF NOT, BRANCH TO CONTINUE

TEST - PMG COUNTER

```

21050 .SBTTL TEST - PMG COUNTER
21051 ;USING 2 Q22BE THIS TEST WILL VALIDATE THAT PMG COUNTER IS REALLY
21052 ;CAPABLE OF GRANTING CPU BUS MASTERSHIP WHEN DMA REQUESTS ARE STILL
21053 ;PENDING.
21054 ;ROUTINE TEST
21055 ;. IF UFD OR NO 2ND Q22BE THEN
21056 ;. EXIT TEST
21057 ;.
21058 ;. ENDF
21059 ;. SET PMG COUNT TO SOME VALUE
21060 ;. PROGRAM BOTH Q22BE TO INTERRUPT AT THE SAME LEVEL
21061 ;. DETERMINE WHICH HAS HIGHER PRIORITY ON THE BUS
21062 ;. PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
21063 ;. PROGRAM 2ND ONE TO DO A CYCLE
21064 ;. CACHE INSTRUCTION THAT WILL STOP 2ND DMA
21065 ;. INITIATE BOTH DMA CYCLES
21066 ;. STOP 2ND DMA (RESET IS "STOLEN" CPU BUS CYCLE)
21067 ;. IF 2ND DMA HAPPENED THEN
21068 ;. ERROR
21069 ;. ENDF
21070 ;. CLEAR PMG COUNT
21071 ;. PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
21072 ;. PROGRAM 2ND ONE TO DO A CYCLE
21073 ;. CACHE INSTRUCTION THAT WILL STOP 2ND DMA
21074 ;. INITIATE BOTH DMA CYCLES
21075 ;. STOP 2ND DMA (CLEARING OF CSR2<0> IS "STOLEN" CPU BUS CYCLE)
21076 ;. IF 2ND DMA DIDN'T HAPPENED THEN
21077 ;. ERROR
21078 ;. ENDF
21079 ;ENDROUTINE
21080 ;*****
TST66: SCOPE
21081 126230 000004 NOP
21082 126232 000240 TST CCHPAS ;have done enough inclusive passes?
21083 126234 005737 003032 BNE 99$ ; nct yet
21084 126242 001003 NOP ; debug aid
21085 126244 000137 126750 JMP 11$ ; yes skip this
21086 126250 000240 99$: NOP
21087
21088 126252 032737 000200 000052 BIT #BIT07,#52 ;UFD MODE?
21089 126260 001402 BEQ 100$ ;IF NOT, CONTINUE
21090 126262 000137 140340 JMP $EOP ;EXIT
21091 126266 005737 002704 100$: TST CSR12 ;SECOND Q22BE AVAILABLE?
21092 126272 001002 BNE 110$ ;IF YES, GO DO TEST
21093 126274 000137 140340 JMP $EOP ;OTHERWISE, GOTO EOP
21094 ;
21095 ; DETERMINE WHICH Q22BE IS CLOSER TO CPU BY DOING DATO
21096 ; THE CSR1 OF THE ONE WITH HIGHER PRIORITY IS IN R4, THE SECOND IN R2
21097 ;
21098 126300 005077 054362 110$: CLR @CSR2 ;CLEAR JUST IN CASE
21099 126304 005077 054376 CLR @CSR22
21100 126310 012777 002740 054352 MOV @TEMP,@BA ;ADDRESS TO BE USED
21101 126316 012777 002740 054364 MOV @TEMP,@BA2 ;FOR BOTH OF THEM
21102 126324 012777 177777 054340 MOV @177777,@WC ;DO FOR 1 WORD
21103 126332 012777 177777 054352 MOV @177777,@WC2 ;IN BOTH Q22BW'S
21104 126340 012777 012525 054326 MOV @12525,@DATA ;DATA FOR 1ST
21105 126346 012777 052525 054340 MOV @52525,@DATA2 ;DATA FOR 2ND

```

TEST - PMG COUNTER

21106	126354	012777	001601	054302	MOV	#1601,BCSR1	;1 DATO FOR 1ST
21107	126362	012777	001601	054314	MOV	#1601,BCSR12	;AND 2ND
21108	126370	012777	000001	054304	MOV	#1,@SIMGOA	;BOTH GO
21109	126376	105777	054262		1\$: TSTB	@CSR1	;FIRST DONE?
21110	126402	100375			BPL	1\$;IF NOT, WAIT
21111	126404	105777	054274		2\$: TSTB	@CSR12	;SECOND DONE
21112	126410	100375			BPL	2\$;WAIT FOR 2ND
21113	126412	022737	052525	002740	CMP	#52525,TEMP	;SECOND FINISHED LAST?
21114	126420	001405			BEQ	3\$;IF SO, BRANCH
21115	126422	013704	002704		MOV	CSR12,R4	;SECOND ONE AT HIGHER LEVEL
21116	126426	013702	002664		MOV	CSR1,R2	;FIRST AT LOWER
21117	126432	000404			BR	4\$;DO PMG PART
21118	126434	013704	002664		3\$: MOV	CSR1,R4	;FIRST ONE AT HIGHER LEVEL
21119	126440	013702	002704		MOV	CSR12,R2	;SECOND AT LOWER
21120							
21121							
21122							
21123	126444	013737	177520	002730	4\$: MOV	BCSR,SAVBR	;STORE BCSR REGISTER
21124	126452	012700	000001		MOV	#BIT00,R0	;FIRST VALUE FOR PMG 0.4msec
21125	126456	050037	177520		5\$: BIS	R0,BCSR	;CHANGE PMG COUNTER
21126	126462	005737	126564		TST	6\$;CACHE RESET INSTRUCTION
21127	126466	005737	126566		TST	6\$+2	;AND THE FOLLOWING FEW
21128	126472	005737	126570		TST	6\$+4	
21129	126476	005737	126572		TST	6\$+6	
21130	126502	012714	001407		MOV	#1407,(R4)	;DATI IN HOG MODE FOR HIGHER ONE
21131	126506	005064	000002		CLR	2(R4)	;CLEAR CSR2
21132	126512	012764	002740	000004	MOV	#TEMP,4(R4)	;ADDRESS TO START DATI
21133	126520	012764	000000	000006	MOV	#0,6(R4)	;DO 128 DATI'S IN HOG MODE
21134	126526	012712	001407		MOV	#1407,(R2)	;DATO FOR LOWER LEVEL ONE
21135	126532	005062	000002		CLR	2(R2)	;CLEAR JUST IN CASE
21136	126536	012762	002740	000004	MOV	#TEMP,4(R2)	;USE THE SAME ADDRESS
21137	126544	012762	177777	000006	MOV	#177777,6(R2)	;DO JUST ONE DATO
21138	126552	005062	000010		CLR	10(R2)	;CLEAR DATA OF 2ND Q22BE
21139	126556	012762	000001	000016	MOV	#1,16(R2)	;BOTH GO
21140	126564	000005			6\$: RESET		;STOP Q22BE AT R4
21141	126566	023762	002740	000010	CMP	TEMP,10(R2)	;SECOND DMA DONE?
21142	126574	001001			BNE	7\$;IF SET, BRANCH
21143	126576	104127			ERROR	+127	;NO CYCLE STEALING
21144	126600	062700	000003		7\$: ADD	#3,R0	;DO FOR 1,3,7 IN PMG
21145	126604	040037	177520		BIC	R0,BCSR	;CLEAR PREVIOUS BITS IN BCSR
21146	126610	022700	000007		CMP	#7,R0	;LAST ONE?
21147	126614	002320			BGE	5\$;IF NOT, BRANCH
21148							
21149							
21150							
21151	126616	042737	000007	177520	BIC	#7,BCSR	;TURN OF PMG COUNTER
21152	126624	005737	126726		TST	8\$;CACHE RESET FETCH
21153	126630	005737	126730		TST	8\$+2	;AND THE FOLLOWING FEW
21154	126634	005737	126732		TST	8\$+4	
21155	126640	005737	126734		TST	8\$+6	
21156	126644	012714	001407		MOV	#1407,(R4)	;DATI IN HOG MODE FOR HIGHER ONE
21157	126650	005064	000002		CLR	2(R4)	;CLEAR CSR2
21158	126654	012764	002740	000004	MOV	#TEMP,4(R4)	;ADDRESS TO START DATI
21159	126662	012764	000000	000006	MOV	#0,6(R4)	;DO 128 DATI'S IN HOG MODE
21160	126670	012712	001407		MOV	#1407,(R2)	;DATO FOR LOWER LEVEL ONE
21161	126674	005062	000002		CLR	2(R2)	;CLEAR CRS2 OF 2ND
21162	126700	012762	002740	000004	MOV	#TEMP,4(R2)	;USE THE SAME ADDRESS

GLOBAL ERROR MESSAGES

21184					.SBTTL GLOBAL ERROR MESSAGES
21185	127002	102	101	123	EM1: .ASCIZ /BASIC INSTRUCTION SET ERROR/
	127005	111	103	040	
	127010	111	116	123	
	127013	124	122	125	
	127016	103	124	111	
	127021	117	116	040	
	127024	123	105	124	
	127027	040	105	122	
	127032	122	117	122	
	127035	000			
21186	127036	115	115	125	EM2: .ASCIZ /MMU ERROR/
	127041	040	105	122	
	127044	122	117	122	
	127047	000			
21187	127050	106	120	120	EM3: .ASCIZ /FPP ERROR/
	127053	040	105	122	
	127056	122	117	122	
	127061	000			
21188	127062	105	122	122	EM4: .ASCIZ /ERROR IN READ-WRITE BITS OF CCR/
	127065	117	122	040	
	127070	111	116	040	
	127073	122	105	101	
	127076	104	055	127	
	127101	122	111	124	
	127104	105	040	102	
	127107	111	124	123	
	127112	040	117	106	
	127115	040	103	103	
	127120	122	000		
21189	127122	106	117	122	EM5: .ASCIZ /FORCE MISS WRITES TO CACHE/
	127125	103	105	040	
	127130	115	111	123	
	127133	123	040	127	
	127136	122	111	124	
	127141	105	123	040	
	127144	124	117	040	
	127147	103	101	103	
	127152	110	105	000	
21190	127155	106	117	122	EM6: .ASCIZ /FORCE MISS WRITE INVALIDATES CACHE/
	127160	103	105	040	
	127163	115	111	123	
	127166	123	040	127	
	127171	122	111	124	
	127174	105	040	111	
	127177	116	126	101	
	127202	114	111	104	
	127205	101	124	105	
	127210	123	040	103	
	127213	101	103	110	
	127216	105	000		
21191	127220	125	116	105	EM7: .ASCIZ /UNEXPECTED PARITY INTERRUPT/
	127223	130	120	105	
	127226	103	124	105	
	127231	104	040	120	
	127234	101	122	111	
	127237	124	131	040	

GLOBAL ERROR MESSAGES

	127242	111	116	124		
	127245	105	122	122		
	127250	125	120	124		
	127253	000				
21192	127254	124	101	107	EM10:	.ASCIZ /TAG PARITY ERROR/
	127257	040	120	101		
	127262	122	111	124		
	127265	131	040	105		
	127270	122	122	117		
	127273	122	000			
21193	127275	104	101	124	EM11:	.ASCIZ /DATA PARITY ERROR/
	127300	101	040	120		
	127303	101	122	111		
	127306	124	131	040		
	127311	105	122	122		
	127314	117	122	000		
21194	127317	114	117	127	EM12:	.ASCIZ /LOW BYTE PARITY ERROR/
	127322	040	102	131		
	127325	124	105	040		
	127330	120	101	122		
	127333	111	124	131		
	127336	040	105	122		
	127341	122	117	122		
	127344	000				
21195	127345	110	111	107	EM13:	.ASCIZ /HIGH BYTE PARITY ERROR/
	127350	110	040	102		
	127353	131	124	105		
	127356	040	120	101		
	127361	122	111	124		
	127364	131	040	105		
	127367	122	122	117		
	127372	122	000			
21196	127374	105	122	122	EM14:	.ASCIZ /ERROR IN DATA PATH/
	127377	117	122	040		
	127402	111	116	040		
	127405	104	101	124		
	127410	101	040	120		
	127413	101	124	110		
	127416	000				
21197	127417	106	117	122	EM15:	.ASCIZ /FORCE MISS READS FROM CACHE/
	127422	103	105	040		
	127425	115	111	123		
	127430	123	040	122		
	127433	105	101	104		
	127436	123	040	106		
	127441	122	117	115		
	127444	040	103	101		
	127447	103	110	105		
	127452	000				
21198	127453	106	117	122	EM16:	.ASCIZ /FORCE MISS READS FROM CACHE AND MISS/
	127456	103	105	040		
	127461	115	111	123		
	127464	123	040	122		
	127467	105	101	104		
	127472	123	040	106		
	127475	122	117	115		
	127500	040	103	101		

GLOBAL ERROR MESSAGES

	127503	103	110	105	
	127506	040	101	116	
	127511	104	040	115	
	127514	111	123	123	
	127517	000			
21199	127520	105	122	122	EM17: .ASCIZ \ERROR IN RECORDING HITS IN HIT/MISS\
	127523	117	122	040	
	127526	111	116	040	
	127531	122	105	103	
	127534	117	122	104	
	127537	111	116	107	
	127542	040	110	111	
	127545	124	123	040	
	127550	111	116	040	
	127553	110	111	124	
	127556	057	115	111	
	127561	123	123	000	
21200	127564	127	122	111	EM20: .ASCIZ /WRITE BYTE ALLOCATES CACHE/
	127567	124	105	040	
	127572	102	131	124	
	127575	105	040	101	
	127600	114	114	117	
	127603	103	101	124	
	127606	105	123	040	
	127611	103	101	103	
	127614	110	105	000	
21201	127617	127	122	111	EM21: .ASCIZ /WRITE BYTE HIT DOES NOT RECORD HIT/
	127622	124	105	040	
	127625	102	131	124	
	127630	105	040	110	
	127633	111	124	040	
	127636	104	117	105	
	127641	123	040	116	
	127644	117	124	040	
	127647	122	105	103	
	127652	117	122	104	
	127655	040	110	111	
	127660	124	000		
21202	127662	102	131	124	EM22: .ASCIZ /BYTES REVERSED ON WRITE CYCLES/
	127665	105	123	040	
	127670	122	105	126	
	127673	105	122	123	
	127676	105	104	040	
	127701	117	116	040	
	127704	127	122	111	
	127707	124	105	040	
	127712	103	131	103	
	127715	114	105	123	
	127720	000			
21203	127721	103	117	116	EM23: .ASCIZ /CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE/
	127724	104	111	124	
	127727	111	117	116	
	127732	101	114	040	
	127735	102	131	120	
	127740	101	123	123	
	127743	040	104	117	
	127746	105	123	116	

GLOBAL ERROR MESSAGES

	127751	047	124	040	
	127754	111	116	126	
	127757	101	114	111	
	127762	104	101	124	
	127765	105	040	103	
	127770	101	103	110	
	127773	105	000		
21204	127775	110	111	124	EM24: .ASCIZ /HITS RECORDED AFTER FLUSHING CACHE/
	130000	123	040	122	
	130003	105	103	117	
	130006	122	104	105	
	130011	104	040	101	
	130014	106	124	105	
	130017	122	040	106	
	130022	114	125	123	
	130025	110	111	116	
	130030	107	040	103	
	130033	101	103	110	
	130036	105	000		
21205	130040	102	131	120	EM25: .ASCIZ /BYPASS DOESN'T INVALIDATE CACHE/
	130043	101	123	123	
	130046	040	104	117	
	130051	105	123	116	
	130054	047	124	040	
	130057	111	116	126	
	130062	101	114	111	
	130065	104	101	124	
	130070	105	040	103	
	130073	101	103	110	
	130076	105	000		
21206	130100	115	123	105	EM26: .ASCIZ /MSER DOES NOT CLEAR ON WRITE REFERENCE/
	130103	122	040	104	
	130106	117	105	123	
	130111	040	116	117	
	130114	124	040	103	
	130117	114	105	101	
	130122	122	040	117	
	130125	116	040	127	
	130130	122	111	124	
	130133	105	040	122	
	130136	105	106	105	
	130141	122	105	116	
	130144	103	105	000	
21207	130147	120	101	122	EM27: .ASCIZ /PARITY ERROR DON'T CAUSE A MISS/
	130152	111	124	131	
	130155	040	105	122	
	130160	122	117	122	
	130163	040	104	117	
	130166	116	047	124	
	130171	040	103	101	
	130174	125	123	105	
	130177	040	101	040	
	130202	115	111	123	
	130205	123	000		
21208	130207	120	101	122	EM30: .ASCIZ /PARITY ERROR DON'T SET MSER WITH CCR<7>=0/
	130212	111	124	131	
	130215	040	105	122	

GLOBAL ERROR MESSAGES

	130220	122	117	122	
	130223	040	104	117	
	130226	116	047	124	
	130231	040	123	105	
	130234	124	040	115	
	130237	123	105	122	
	130242	040	127	111	
	130245	124	110	040	
	130250	103	103	122	
	130253	074	067	076	
	130256	075	060	000	
21209	130261	120	101	122	EM31: .ASCIZ /PARITY ERROR IGNORED/
	130264	111	124	131	
	130267	040	105	122	
	130272	122	117	122	
	130275	040	111	107	
	130300	116	117	122	
	130303	105	104	000	
21210	130306	120	101	122	EM32: .ASCIZ /PARITY ERROR IGNORED ON LOW BYTE/
	130311	111	124	131	
	130314	040	105	122	
	130317	122	117	122	
	130322	040	111	107	
	130325	116	117	122	
	130330	105	104	040	
	130333	117	116	040	
	130336	114	117	127	
	130341	040	102	131	
	130344	124	105	000	
21211	130347	120	101	122	EM33: .ASCIZ /PARITY ERROR IGNORED ON HIGH BYTE/
	130352	111	124	131	
	130355	040	105	122	
	130360	122	117	122	
	130363	040	111	107	
	130366	116	117	122	
	130371	105	104	040	
	130374	117	116	040	
	130377	110	111	107	
	130402	110	040	102	
	130405	131	124	105	
	130410	000			
21212	130411	120	101	122	EM34: .ASCIZ /PARITY ABORT LOGIC DOESN'T WORK/
	130414	111	124	131	
	130417	040	101	102	
	130422	117	122	124	
	130425	040	114	117	
	130430	107	111	103	
	130433	040	104	117	
	130436	105	123	116	
	130441	047	124	040	
	130444	127	117	122	
	130447	113	000		
21213	130451	115	123	105	EM35: .ASCIZ /MSER NOT SET PROPERLY/
	130454	122	040	116	
	130457	117	124	040	
	130462	123	105	124	
	130465	040	120	122	

GLOBAL ERROR MESSAGES

	130470	117	120	105	
	130473	122	114	131	
	130476	000			
21214	130477	120	101	122	EM36: .ASCIZ /PARITY INTERRUPT LOGIC DOESN'T WORK/
	130502	111	124	131	
	130505	040	111	116	
	130510	124	105	122	
	130513	122	125	120	
	130516	124	040	114	
	130521	117	107	111	
	130524	103	040	104	
	130527	117	105	123	
	130532	116	047	124	
	130535	040	127	117	
	130540	122	113	000	
21215	130543	116	130	115	EM37: .ASCIZ /NXM AND PARITY ABORT DIN'T HAPPEN/
	130546	040	101	116	
	130551	104	040	120	
	130554	101	122	111	
	130557	124	131	040	
	130562	101	102	117	
	130565	122	124	040	
	130570	104	111	116	
	130573	047	124	040	
	130576	110	101	120	
	130601	120	105	116	
	130604	000			
21216	130605	120	101	122	EM40: .ASCIZ /PARITY ABORT NOT BLOCKED BY NXM TRAP/
	130610	111	124	131	
	130613	040	101	102	
	130616	117	122	124	
	130621	040	116	117	
	130624	124	040	102	
	130627	114	117	103	
	130632	113	105	104	
	130635	040	102	131	
	130640	040	116	130	
	130643	115	040	124	
	130646	122	101	120	
	130651	000			
21217	130652	115	125	114	EM41: .ASCIZ /MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS/
	130655	124	111	055	
	130660	120	122	117	
	130663	103	105	123	
	130666	123	117	122	
	130671	040	110	117	
	130674	117	113	040	
	130677	111	116	123	
	130702	124	122	125	
	130705	103	124	111	
	130710	117	116	040	
	130713	104	117	105	
	130716	123	116	047	
	130721	124	040	103	
	130724	101	125	123	
	130727	105	040	115	
	130732	111	123	123	

GLOBAL ERRCR MESSAGES

	130735	000				
21218	130736	105	122	122	EM42:	.ASCIZ /ERROR IN PARITY LOGIC/
	130741	117	122	040		
	130744	111	116	040		
	130747	120	101	122		
	130752	111	124	131		
	130755	040	114	117		
	130760	107	111	103		
	130763	000				
21219	130764	105	122	122	EM43:	.ASCIZ /ERROR IN CACHE DATA RAMS/
	130767	117	122	040		
	130772	111	116	040		
	130775	103	101	103		
	131000	110	105	040		
	131003	104	101	124		
	131006	101	040	122		
	131011	101	115	123		
	131014	000				
21220	131015	105	122	122	EM44:	.ASCIZ /ERROR IN NXM IN STANDALONE MODE/
	131020	117	122	040		
	131023	111	116	040		
	131026	116	130	115		
	131031	040	111	116		
	131034	040	123	124		
	131037	101	116	104		
	131042	101	114	117		
	131045	116	105	040		
	131050	115	117	104		
	131053	105	000			
21221	131055	105	122	122	EM45:	.ASCIZ \ERROR IN RECORDING HITS THROUGH HIT/MISS REGISTER\
	131060	117	122	040		
	131063	111	116	040		
	131066	122	105	103		
	131071	117	122	104		
	131074	111	116	107		
	131077	040	110	111		
	131102	124	123	040		
	131105	124	110	122		
	131110	117	125	107		
	131113	110	040	110		
	131116	111	124	057		
	131121	115	111	123		
	131124	123	040	122		
	131127	105	107	111		
	131132	123	124	105		
	131135	122	000			
21222	131137	110	111	124	EM46:	.ASCIZ /HIT RECORDED FOR A LOCATION THAT SHOULD NOT BE IN CACHE/
	131142	040	122	105		
	131145	103	117	122		
	131150	104	105	104		
	131153	040	106	117		
	131156	122	040	101		
	131161	040	114	117		
	131164	103	101	124		
	131167	111	117	116		
	131172	040	124	110		
	131175	101	124	040		

GLOBAL ERROR MESSAGES

	131200	123	110	117	
	131203	125	114	104	
	131206	040	116	117	
	131211	124	040	102	
	131214	105	040	111	
	131217	116	040	103	
	131222	101	103	110	
	131225	105	000		
21223	131227	115	111	123	EM47: .ASCIZ /MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE/
	131232	123	040	122	
	131235	105	103	117	
	131240	122	104	105	
	131243	104	040	106	
	131246	117	122	040	
	131251	101	040	114	
	131254	117	103	101	
	131257	124	111	117	
	131262	116	040	124	
	131265	110	101	124	
	131270	040	123	110	
	131273	117	125	114	
	131276	104	040	102	
	131301	105	040	111	
	131304	116	040	103	
	131307	101	103	110	
	131312	105	000		
21224	131314	105	122	122	EM50: .ASCIZ /ERROR IN TAG STORE/
	131317	117	122	040	
	131322	111	116	040	
	131325	124	101	107	
	131330	040	123	124	
	131333	117	122	105	
	131336	000			
21225	131337	105	122	122	EM51: .ASCIZ /ERROR PCR READ-WRITE BITS/
	131342	117	122	040	
	131345	120	103	122	
	131350	040	122	105	
	131353	101	104	055	
	131356	127	122	111	
	131361	124	105	040	
	131364	102	111	124	
	131367	123	000		
21226	131371	105	122	122	EM52: .ASCIZ /ERROR IN BCSR READ WRITE BITS/
	131374	117	122	040	
	131377	111	116	040	
	131402	102	103	123	
	131405	122	040	122	
	131410	105	101	104	
	131413	055	127	122	
	131416	111	124	105	
	131421	040	102	111	
	131424	124	123	000	
21227	131427	122	105	123	EM53: .ASCIZ /RESET DOESN'T CLEAR BCSR<4>/
	131432	105	124	040	
	131435	104	117	105	
	131440	123	116	047	
	131443	124	040	103	

GLOBAL ERROR MESSAGES

	131446	114	105	101	
	131451	122	040	102	
	131454	103	123	122	
	131457	074	064	076	
	131462	000			
21228	131463	103	110	105	EM54: .ASCIZ /CHECKSUM ERROR IN 16 BIT ROM /
	131466	103	113	123	
	131471	125	115	040	
	131474	105	122	122	
	131477	117	122	040	
	131502	111	116	040	
	131505	061	066	055	
	131510	102	111	124	
	131513	040	122	117	
	131516	115	040	000	
21229	131521	103	110	105	EM55: .ASCIZ /CHECKSUM ERROR IN 8 BIT ROM/
	131524	103	113	123	
	131527	125	115	040	
	131532	105	122	122	
	131535	117	122	040	
	131540	111	116	040	
	131543	070	055	102	
	131546	111	124	040	
	131551	122	117	115	
	131554	000			
21230	131555	124	111	115	EM56: .ASCIZ /TIMEOUT READING LKS/
	131560	105	117	125	
	131563	124	040	122	
	131566	105	101	104	
	131571	111	116	107	
	131574	040	114	113	
	131577	123	000		
21231	131601	114	113	123	EM57: .ASCIZ /LKS<07> DOES NOT BECOME 1/
	131604	074	060	067	
	131607	076	040	104	
	131612	117	105	123	
	131615	040	116	117	
	131620	124	040	102	
	131623	105	103	117	
	131626	115	105	040	
	131631	061	000		
21232	131633	127	122	111	EM60: .ASCIZ /WRITE REFERENCE DOESN'T CLEAR LKS<07>/
	131636	124	105	040	
	131641	122	105	106	
	131644	105	122	105	
	131647	116	103	105	
	131652	040	104	117	
	131655	105	123	116	
	131660	047	124	040	
	131663	103	114	105	
	131666	101	122	040	
	131671	114	113	123	
	131674	074	060	067	
	131677	076	000		
21233	131701	111	114	114	EM61: .ASCIZ /ILLEGAL LKS INTERRUPTS/
	131704	105	107	101	
	131707	114	040	114	

GLOBAL ERROR MESSAGES

	131712	113	123	040	
	131715	111	116	124	
	131720	105	122	122	
	131723	125	120	124	
	131726	123	000		
21234	131730	120	122	117	EM62: .ASCIZ /PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>/
	131733	103	105	123	
	131736	123	117	122	
	131741	040	111	116	
	131744	124	105	122	
	131747	122	125	120	
	131752	124	123	040	
	131755	104	117	116	
	131760	047	124	040	
	131763	103	114	105	
	131766	101	122	040	
	131771	114	113	123	
	131774	074	060	067	
	131777	076	000		
21235	132001	114	113	123	EM63: .ASCIZ /LKS READY DOESN'T GO LOW/
	132004	040	122	105	
	132007	101	104	131	
	132012	040	104	117	
	132015	105	123	116	
	132020	047	124	040	
	132023	107	117	040	
	132026	114	117	127	
	132031	000			
21236	132032	127	122	117	EM64: .ASCIZ /WRONG NUMBER OF LKS INTERRUPTS/
	132035	116	107	040	
	132040	116	125	115	
	132043	102	105	122	
	132046	040	117	106	
	132051	040	114	113	
	132054	123	040	111	
	132057	116	124	105	
	132062	122	122	125	
	132065	120	124	123	
	132070	000			
21237	132071	114	113	123	EM65: .ASCIZ /LKS INTERRUPTS HAPPEN AT WRONG PRIORITY/
	132074	040	111	116	
	132077	124	105	122	
	132102	122	125	120	
	132105	124	123	040	
	132110	110	101	120	
	132113	120	105	116	
	132116	040	101	124	
	132121	040	127	122	
	132124	117	116	107	
	132127	040	120	122	
	132132	111	117	122	
	132135	111	124	131	
	132140	000			
21238	132141	102	103	123	EM66: .ASCIZ /BCSR<12> DOES NOT DISABLES LKS/
	132144	122	074	061	
	132147	062	076	040	
	132152	104	117	105	

GLOBAL ERROR MESSAGES

	132155	123	040	116	
	132160	117	124	040	
	132163	104	111	123	
	132166	101	102	114	
	132171	105	123	040	
	132174	114	113	123	
	132177	000			
21239	132200	102	103	123	EM67: .ASCIZ /BCSR<13> DOESN'T SET LKS<06>/
	132203	122	074	061	
	132206	063	076	040	
	132211	104	117	105	
	132214	123	116	047	
	132217	124	040	123	
	132222	105	124	040	
	132225	114	113	123	
	132230	074	060	066	
	132233	076	000		
21240	132235	122	105	123	EM70: .ASCIZ /RESET DOESN'T SET LKS<7>/
	132240	105	124	040	
	132243	104	117	105	
	132246	123	116	047	
	132251	124	040	123	
	132254	105	124	040	
	132257	114	113	123	
	132262	074	067	076	
	132265	000			
21241	132266	122	105	123	EM71: .ASCIZ /RESET DOESN'T CLEAR LKS<06>/
	132271	105	124	040	
	132274	104	117	105	
	132277	123	116	047	
	132302	124	040	103	
	132305	114	105	101	
	132310	122	040	114	
	132313	113	123	074	
	132316	060	066	076	
	132321	000			
21242	132322	124	111	115	EM72: .ASCIZ /TIMEOUT READING SLU REGISTERS/
	132325	105	117	125	
	132330	124	040	122	
	132333	105	101	104	
	132336	111	116	107	
	132341	040	123	114	
	132344	125	040	122	
	132347	105	107	111	
	132352	123	124	105	
	132355	122	123	000	
21243	132360	105	122	122	EM73: .ASCIZ /ERROR IN XMIT READY/
	132363	117	122	040	
	132366	111	116	040	
	132371	130	115	111	
	132374	104	040	122	
	132377	105	101	104	
	132402	131	000		
21244	132404	122	103	123	EM74: .ASCIZ /RCSR<7> DOESN'T BECOME 1/
	132407	122	074	067	
	132412	076	040	104	
	132415	117	105	123	

GLOBAL ERROR MESSAGES

	132420	116	047	124	
	132423	040	102	105	
	132426	103	117	115	
	132431	105	040	061	
	132434	000			
21245	132435	127	122	117	EM75: .ASCIZ /WRONG CHARACTER RECEIVED/
	132440	116	107	040	
	132443	103	110	101	
	132446	122	101	103	
	132451	124	105	122	
	132454	040	122	105	
	132457	103	105	111	
	132462	126	105	104	
	132465	000			
21246	132466	122	103	123	EM76: .ASCIZ /RCSR<07> NOT CLEARED AFTER READING RBUF/
	132471	122	074	060	
	132474	067	076	040	
	132477	116	117	124	
	132502	040	103	114	
	132505	105	101	122	
	132510	105	104	040	
	132513	101	106	124	
	132516	105	122	040	
	132521	122	105	101	
	132524	104	111	116	
	132527	107	040	122	
	132532	102	125	106	
	132535	000			
21247	132536	130	103	123	EM77: .ASCIZ /XCSR<07> NOT SET ON RESET/
	132541	122	074	060	
	132544	067	076	040	
	132547	116	117	124	
	132552	040	123	105	
	132555	124	040	117	
	132560	116	040	122	
	132563	105	123	105	
	132566	124	000		
21248	132570	122	103	123	EM100: .ASCIZ /RCSR<07> NOT CLEARED ON RESET/
	132573	122	074	060	
	132576	067	076	040	
	132601	116	117	124	
	132604	040	103	114	
	132607	105	101	122	
	132612	105	104	040	
	132615	117	116	040	
	132620	122	105	123	
	132623	105	124	000	
21249	132626	123	114	125	EM101: .ASCIZ /SLU INTERRUPTS HAPPEN AT 4/
	132631	040	111	116	
	132634	124	105	122	
	132637	122	125	120	
	132642	124	123	040	
	132645	110	101	120	
	132650	120	105	116	
	132653	040	101	124	
	132656	040	064	000	
21250	132661	122	105	123	EM102: .ASCIZ /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/

GLOBAL ERROR MESSAGES

	132664	105	124	040	
	132667	104	117	105	
	132672	123	040	116	
	132675	117	124	040	
	132700	103	114	105	
	132703	101	122	040	
	132706	120	122	117	
	132711	120	105	122	
	132714	040	102	111	
	132717	124	123	040	
	132722	111	116	040	
	132725	123	114	125	
	132730	040	122	105	
	132733	107	111	123	
	132736	124	105	122	
	132741	123	000		
21251	132743	124	122	101	EM103: .ASCIZ /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07 /
	132746	116	123	115	
	132751	111	124	040	
	132754	111	116	124	
	132757	105	122	122	
	132762	125	120	124	
	132765	040	104	117	
	132770	105	123	040	
	132773	116	117	124	
	132776	040	103	114	
	133001	105	101	122	
	133004	040	130	103	
	133007	123	122	074	
	133012	060	067	076	
	133015	000			
21252	133016	122	105	103	EM104: .ASCIZ /RECEIVE INTERRUPTS DGN'T CLEAR RCSR<07>/
	133021	105	111	126	
	133024	105	040	111	
	133027	116	124	105	
	133032	122	122	125	
	133035	120	124	123	
	133040	040	104	117	
	133043	116	047	124	
	133046	040	103	114	
	133051	105	101	122	
	133054	040	122	103	
	133057	123	122	074	
	133062	060	067	076	
	133065	000			
21253	133066	102	122	105	EM105: .ASCIZ /BREAK CONDITION DOES NOT SET RBUF PROPERLY/
	133071	101	113	040	
	133074	103	117	116	
	133077	104	111	124	
	133102	111	117	116	
	133105	040	104	117	
	133110	105	123	040	
	133113	116	117	124	
	133116	040	123	105	
	133121	124	040	122	
	133124	102	125	106	
	133127	040	120	122	

GLOBAL ERROR MESSAGES

	133132	117	120	105	
	133135	122	114	131	
	133140	000			
21254	133141	122	102	125	EM106: .ASCIZ /RBUF <15-11> WASN'T CLEARED ON NEXT CHARACTER/
	133144	106	040	074	
	133147	061	065	055	
	133152	061	061	076	
	133155	040	127	101	
	133160	123	116	047	
	133163	124	040	103	
	133166	114	105	101	
	133171	122	105	104	
	133174	040	117	116	
	133177	040	116	105	
	133202	130	124	040	
	133205	103	110	101	
	133210	122	101	103	
	133213	124	105	122	
	133216	000			
21255	133217	123	114	125	EM107: .ASCIZ /SLU INTERRUPTS DON'T HAPPEN/
	133222	040	111	116	
	133225	124	105	122	
	133230	122	125	120	
	133233	124	123	040	
	133236	104	117	116	
	133241	047	124	040	
	133244	110	101	120	
	133247	120	105	116	
	133252	000			
21256	133253	105	122	122	EM110: .ASCIZ /ERROR IN WRITING TO SLU REGISTERS/
	133256	117	122	040	
	133261	111	116	040	
	133264	127	122	111	
	133267	124	111	116	
	133272	107	040	124	
	133275	117	040	123	
	133300	114	125	040	
	133303	122	105	107	
	133306	111	123	124	
	133311	105	122	123	
	133314	000			
21257	133315	106	111	122	EM111: .ASCIZ /FIPST CHARACTER WAS NOT OVERRUN BY THE SECOND/
	133320	123	124	040	
	133323	103	110	101	
	133326	122	101	103	
	133331	124	105	122	
	133334	040	127	101	
	133337	123	040	116	
	133342	117	124	040	
	133345	117	126	105	
	133350	122	122	125	
	133353	116	040	102	
	133356	131	040	124	
	133361	110	105	040	
	133364	123	105	103	
	133367	117	116	104	
	133372	000			

GLOBAL ERROR MESSAGES

21258	133373	117	126	105	EM112: .ASCIZ /OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF/
	133376	122	122	125	
	133401	116	040	103	
	133404	117	116	104	
	133407	111	124	111	
	133412	117	116	040	
	133415	104	117	105	
	133420	123	040	116	
	133423	117	124	040	
	133426	123	105	124	
	133431	040	120	122	
	133434	117	120	105	
	133437	122	040	102	
	133442	111	124	123	
	133445	040	111	116	
	133450	040	122	102	
	133453	125	106	000	
21259	133456	117	126	105	EM113: .ASCIZ /OVERRUN BITS WERE NOT CLEARED ON THE NEXT CHARACTER/
	133461	122	122	125	
	133464	116	040	102	
	133467	111	124	123	
	133472	040	127	105	
	133475	122	105	040	
	133500	116	117	124	
	133503	040	103	114	
	133506	105	101	122	
	133511	105	104	040	
	133514	117	116	040	
	133517	124	110	105	
	133522	040	116	105	
	133525	130	124	040	
	133530	103	110	101	
	133533	122	101	103	
	133536	124	105	122	
	133541	000			
21260	133542	105	122	122	EM114: .ASCIZ \ERROR ON XCSR<2>\
	133545	117	122	040	
	133550	117	116	040	
	133553	130	103	123	
	133556	122	074	062	
	133561	076	000		
21261	133563	105	122	122	EM115: .ASCIZ /ERROR IN TAG STORE FROM STANDALONE MODE/
	133566	117	122	040	
	133571	111	116	040	
	133574	124	101	107	
	133577	040	123	124	
	133602	117	122	105	
	133605	040	106	122	
	133610	117	115	040	
	133613	123	124	101	
	133616	116	104	101	
	133621	114	117	116	
	133624	105	040	115	
	133627	117	104	105	
	133632	000			
21262	133633	104	115	101	EM116: .ASCIZ /DMA TAG PARITY DOES NOT GENERATE PROPER RESPONSE/
	133636	040	124	101	

GLOBAL ERROR MESSAGES

	133641	107	040	120	
	133644	101	122	111	
	133647	124	131	040	
	133652	104	117	105	
	133655	123	040	116	
	133660	117	124	040	
	133663	107	105	116	
	133666	105	122	101	
	133671	124	105	040	
	133674	120	122	117	
	133677	120	105	122	
	133702	040	122	105	
	133705	123	120	117	
	133710	116	123	105	
	133713	000			
21263	133714	115	123	105	EM117: .ASCIZ /MSER<13>NOT SET IN STANDALONE MODE/
	133717	122	074	061	
	133722	063	076	116	
	133725	117	124	040	
	133730	123	105	124	
	133733	040	111	116	
	133736	040	123	124	
	133741	101	116	104	
	133744	101	114	117	
	133747	116	105	040	
	133752	115	117	104	
	133755	105	000		
21264	133757	104	115	101	EM120: .ASCIZ /DMA WRITE HITS DON'T INVALIDATE CACHE/
	133762	040	127	122	
	133765	111	124	105	
	133770	040	110	111	
	133773	124	123	040	
	133776	104	117	116	
	134001	047	124	040	
	134004	111	116	126	
	134007	101	114	111	
	134012	104	101	124	
	134015	105	040	103	
	134020	101	103	110	
	134023	105	000		
21265	134025	111	116	040	EM121: .ASCIZ /IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED/
	134030	102	114	117	
	134033	103	113	040	
	134036	115	117	104	
	134041	105	040	117	
	134044	116	040	127	
	134047	122	111	124	
	134052	105	040	104	
	134055	115	101	040	
	134060	110	111	124	
	134063	123	040	116	
	134066	117	124	040	
	134071	105	126	105	
	134074	122	131	124	
	134077	110	111	116	
	134102	107	040	111	
	134105	123	040	111	

GLOBAL ERROR MESSAGES

	134110	116	126	101	
	134113	114	111	104	
	134116	101	124	105	
	134121	104	000		
21266	134123	122	105	101	EM122: .ASCIZ /READ DMA HIT IS WRONG/
	134126	104	040	104	
	134131	115	101	040	
	134134	110	111	124	
	134137	040	111	123	
	134142	040	127	122	
	134145	117	116	107	
	134150	000			
21267	134151	105	122	122	EM123: .ASCIZ /ERROR IN Q22BE DMA CYCLES/
	134154	117	122	040	
	134157	111	116	040	
	134162	121	062	062	
	134165	102	105	040	
	134170	104	115	101	
	134173	040	103	131	
	134176	103	114	105	
	134201	123	000		
21268	134203	120	111	122	EM124: .ASCIZ /PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS/
	134206	121	040	111	
	134211	116	124	105	
	134214	122	122	125	
	134217	120	124	123	
	134222	040	104	117	
	134225	116	047	124	
	134230	040	124	101	
	134233	113	105	040	
	134236	120	122	111	
	134241	117	122	111	
	134244	124	131	040	
	134247	117	126	105	
	134252	122	040	121	
	134255	040	102	125	
	134260	123	040	111	
	134263	116	124	105	
	134266	122	122	125	
	134271	120	124	123	
	134274	000			
21269	134275	116	117	040	EM125: .ASCIZ /NO POWER DOWN TRAP TO 24 OCCUR/
	134300	120	117	127	
	134303	105	122	040	
	134306	104	117	127	
	134311	116	040	124	
	134314	122	101	120	
	134317	040	124	117	
	134322	040	062	064	
	134325	040	117	103	
	134330	103	125	122	
	134333	000			
21270	134334	105	122	122	EM126: .ASCIZ /ERROR DOING Q22BE INTERRUPTS/
	134337	117	122	040	
	134342	104	117	111	
	134345	116	107	040	
	134350	121	062	062	

GLOBAL ERROR MESSAGES

	134353	102	105	040	
	134356	111	116	124	
	134361	105	122	122	
	134364	125	120	124	
	134367	123	000		
21271	134371	105	122	122	EM127: .ASCIZ /ERROR IN OPERATION OF PMG COUNTER/
	134374	117	122	040	
	134377	111	116	040	
	134402	117	120	105	
	134405	122	101	124	
	134410	111	117	116	
	134413	040	117	106	
	134416	040	120	115	
	134421	107	040	103	
	134424	117	125	116	
	134427	124	105	122	
	134432	000			
21272	134433	125	116	105	EM130: .ASCIZ /UNEXPECTED TRAP TO 4/
	134436	130	120	105	
	134441	103	124	105	
	134444	104	040	124	
	134447	122	101	120	
	134452	040	124	117	
	134455	040	064	000	
21273	134460	105	122	122	EM131: .ASCIZ /ERROR WRITING TO LKS<6>/
	134463	117	122	040	
	134466	127	122	111	
	134471	124	111	116	
	134474	107	040	124	
	134477	117	040	114	
	134502	113	123	074	
	134505	066	076	000	
21274	134510	105	122	122	EM132: .ASCIZ /ERROR IN MAINTENANCE REGISTER/
	134513	117	122	040	
	134516	111	116	040	
	134521	115	101	111	
	134524	116	124	105	
	134527	116	101	116	
	134532	103	105	040	
	134535	122	105	107	
	134540	111	123	124	
	134543	105	122	000	
21275	134546	105	105	122	EM133: .ASCIZ /EEROM TYPE ERROR/
	134551	117	115	040	
	134554	124	131	120	
	134557	105	040	105	
	134562	122	122	117	
	134565	122	000		
21276	134567	122	105	101	EM134: .ASCIZ /READ OR WRITE EEROM ERROR/
	134572	104	040	117	
	134575	122	040	127	
	134600	122	111	124	
	134603	105	040	105	
	134606	105	122	117	
	134611	115	040	105	
	134614	122	122	117	
	134617	122	000		

GLOBAL ERROR MESSAGES

21277												
21278	134621	040	124	105	DH1:	.ASCII	/	TEST	ERROR/<15><12>			
	134624	123	124	011								
	134627	105	122	122								
	134632	117	122	015								
	134635	012										
21279	134636	040	040	043		.ASCIZ	/	#	PC/			
	134641	011	040	120								
	134644	103	000									
21280	134646	040	124	105	DH4:	.ASCII	/	TEST	ERROR	EXPC TED RECEIVED/<15><12>		
	134651	123	124	011								
	134654	105	122	122								
	134657	117	122	011								
	134662	105	130	120								
	134665	103	124	105								
	134670	104	011	122								
	134673	105	103	105								
	134676	111	126	105								
	134701	104	015	012								
21281	134704	040	040	043		.ASCIZ	/	#	PC	DATA	DATA/	
	134707	011	040	120								
	134712	103	011	040								
	134715	104	101	124								
	134720	101	040	040								
	134723	040	040	040								
	134726	104	101	124								
	134731	101	000									
21282	134733	040	124	105	DH5:	.ASCII	/	TEST	ERROR	HITMIS	DATA IN DATA IN/<15><12>	
	134736	123	124	011								
	134741	105	122	122								
	134744	117	122	011								
	134747	110	111	124								
	134752	115	111	123								
	134755	011	104	101								
	134760	124	101	040								
	134763	111	116	011								
	134766	104	101	124								
	134771	101	040	111								
	134774	116	015	012								
21283	134777	040	040	043		.ASCIZ	/	#	PC	REG.	CACHE	MEMORY/
	135002	011	040	120								
	135005	103	011	040								
	135010	122	105	107								
	135013	056	011	103								
	135016	101	103	110								
	135021	105	011	115								
	135024	105	115	117								
	135027	122	131	000								
21284	135032	040	124	105	DH7:	.ASCII	/	TEST	ERROR	ADDRESS	MSER/<15><12>	
	135035	123	124	011								
	135040	105	122	122								
	135043	117	122	011								
	135046	101	104	104								
	135051	122	105	123								
	135054	123	011	115								
	135057	123	105	122								
	135062	015	012									

GLOBAL ERROR MESSAGES

21285	135064	040	040	043	.ASCIZ / #	PC	ACCESSED/
	135067	011	040	120			
	135072	103	011	101			
	135075	103	103	105			
	135100	123	123	105			
	135103	104	000				
21286	135105	040	124	105	DH24: .ASCII / TEST	ERROR	NUMBER/<15><12>
	135110	123	124	011			
	135113	105	122	122			
	135116	117	122	011			
	135121	116	125	115			
	135124	102	105	122			
	135127	015	012				
21287	135131	040	040	043	.ASCIZ / #	PC	OF HITS/
	135134	011	040	120			
	135137	103	011	117			
	135142	106	040	110			
	135145	111	124	123			
	135150	000					
21288	135151	105	122	122	DH27: .ASCII \ERROR	ERROR	MSER HIT/MISS\<15><12>
	135154	117	122	011			
	135157	105	122	122			
	135162	117	122	011			
	135165	115	123	105			
	135170	122	011	110			
	135173	111	124	057			
	135176	115	111	123			
	135201	123	015	012			
21289	135204	040	040	043	.ASCIZ / #	PC/	
	135207	011	040	120			
	135212	103	000				
21290	135214	040	124	105	DH41: .ASCII / TEST	ERROR	INSTRUCTION/<15><12>
	135217	123	124	011			
	135222	105	122	122			
	135225	117	122	011			
	135230	111	116	123			
	135233	124	122	125			
	135236	103	124	111			
	135241	117	116	015			
	135244	012					
21291	135245	040	040	043	.ASCIZ / #	PC	OPCODE/
	135250	011	040	120			
	135253	103	011	040			
	135256	117	120	103			
	135261	117	104	105			
	135264	000					
21292	135265	040	124	105	DH43: .ASCII / TEST	ERROR	EXPECTD RECEIVD CACHE/<15><12>
	135270	123	124	011			
	135273	105	122	122			
	135276	117	122	011			
	135301	105	130	120			
	135304	105	103	124			
	135307	104	011	122			
	135312	105	103	105			
	135315	111	126	104			
	135320	011	103	101			
	135323	103	110	105			

GLOBAL ERROR MESSAGES

Line	Address	015	012	043	043	PC	DATA	DATA	LOCATION/		
21293	135326	015	012	043							
	135330	040	040	043							
	135333	011	040	120							
	135336	103	011	040							
	135341	104	101	124							
	135344	101	011	040							
	135347	104	101	124							
	135352	101	011	114							
	135355	117	103	101							
	135360	124	111	117							
	135363	116	000								
21294	135365	040	124	105	DH47:	.ASCII	/	TEST	ERROR	ADDRESS	ADDRESS/<15><12>
	135370	123	124	011							
	135373	105	122	122							
	135376	117	122	011							
	135401	101	104	104							
	135404	122	105	123							
	135407	123	011	101							
	135412	104	104	122							
	135415	105	123	123							
	135420	015	012								
21295	135422	040	040	043		.ASCIZ	/	*	PC	<21-16>	<15-0>/
	135425	011	040	120							
	135430	103	011	074							
	135433	062	061	055							
	135436	061	066	076							
	135441	011	040	074							
	135444	061	065	055							
	135447	060	076	000							
21296	135452	040	124	105	DH65:	.ASCII	/	TEST	ERROR	PRIORITY/<15><12>	
	135455	123	124	011							
	135460	105	122	122							
	135463	117	122	011							
	135466	120	122	111							
	135471	117	122	111							
	135474	124	131	015							
	135477	012									
21297	135500	040	040	043		.ASCIZ	/	*	PC	LEVEL/	
	135503	011	040	120							
	135506	103	011	114							
	135511	105	126	105							
	135514	114	000								
21298	135516	040	124	105	DH72:	.ASCII	/	TEST	ERROR	ADDRESS/<15><12>	
	135521	123	124	011							
	135524	105	122	122							
	135527	117	122	011							
	135532	101	104	104							
	135535	122	105	123							
	135540	123	015	012							
21299	135543	040	040	043		.ASCIZ	/	*	PC	FAILED/	
	135546	011	040	120							
	135551	103	011	106							
	135554	101	111	114							
	135557	105	104	000							
21300	135562	040	124	105	DH105:	.ASCII	/	TEST	ERROR	RBUF/<15><12>	
	135565	123	124	011							
	135570	105	122	122							

GLOBAL ERROR MESSAGES

21309	136034	001162	001116	000001	DT4:	.WORD	\$TMP1,\$ERRPC,R1,CCR,0
	136042	177746	000000				
21310	136046	001162	001116	000002	DT5:	.WORD	\$TMP1,\$ERRPC,R2,R1,\$GDDAT,0
	136054	000001	001124	000000			
21311	136062	001162	001116	001122	DT7:	.WORD	\$TMP1,\$ERRPC,\$BDADR,MSER,0
	136070	177744	000000				
21312	136074	001162	001116	001124	DT14:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,TSTLOC,0
	136102	003162	000000				
21313	136106	001162	001116	001124	DT17:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,RECDAT,0
	136114	114150	000000				
21314	136120	001162	001116	000003	DT24:	.WORD	\$TMP1,\$ERRPC,R3,0
	136126	000000					
21315	136130	001162	001116	177744	DT27:	.WORD	\$TMP1,\$ERRPC,MSER,R3,0
	136136	000003	000000				
21316	136142	001162	001116	001124	DT35:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,MSER,0
	136150	177744	000000				
21317	136154	001162	001116	001126	DT41:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,0
	136162	000000					
21318	136164	001162	001116	000001	DT43:	.WORD	\$TMP1,\$ERRPC,R1,RECDAT,\$BDADR,0
	136172	114150	001122	000000			
21319	136200	001162	001116	172354	DT47:	.WORD	\$TMP1,\$ERRPC,KIPAR6,\$BDADR,0
	136206	001122	000000				
21320	136212	001162	001116	000001	DT50:	.WORD	\$TMP1,\$ERRPC,R1,\$BDADR,0
	136220	001122	000000				
21321	136224	001162	001116	001124	DT51:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,PCR,0
	136232	177522	000000				
21322	136236	001162	001116	001124	DT52:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,BCSR,0
	136244	177520	000000				
21323	136250	001162	001116	001124	DT64:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,LKSFL,0
	136256	002722	000000				
21324	136262	001162	001116	001124	DT65:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,0
	136270	000000					
21325	136272	001162	001116	001124	DT75:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,0
	136300	001126	000000				
21326	136304	001162	001116	177562	DT105:	.WORD	\$TMP1,\$ERRPC,RBUF,0
	136312	000000					
21327	136314	001162	001116	001126	DT115:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,KIPAR6,\$BDADR,0
	136322	172354	001122	000000			
21328	136330	001162	001122	000000	DT130:	.WORD	\$TMP1,\$BDADR,0
21329	136336	001162	001116	003022	DT134:	.WORD	\$TMP1,\$ERRPC,ERRCNT,R3,\$BDDAT,R4,\$BDADR,0
	136344	000003	001126	000004			
	136352	001122	000000				

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

21331 .SBTTL MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
21332 ;*****
21333 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
21334 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
21335 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
21336 ;*
21337 ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
21338 ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
21339 ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
21340 ;*FORMAT. RO SHOULD NOT BE USED.
21341
21342 136356          ERTYPE:
21343 136356 005037 001162          CLR      $TMP1          ;;JUST CLEAR IT
21344 136362 113737 001102 001162  MOVB     $TSTNM,$TMP1  ;;STORE TEST NUMBER
21345 136370 104401 001175          TYPE     .$CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
21346 136374 010046          MOV      RO,(SP)      ;;SAVE RO
21347 136376 005000          CLR      RO          ;;PICKUP THE ITEM INDEX
21348 136400 153700 001114  BISB     @#$ITEMB,RO
21349 136404 001004          BNE     1$          ;;IF ITEM NUMBER IS ZERO, JUST
21350                                ;;TYPE THE PC OF THE ERROR
21351 136406 013746 001116  MOV      $ERRPC,-(SP)  ;;SAVE $ERRPC FOR TYPEOUT
21352                                ;;ERROR ADDRESS
21353 136412 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
21354 136414 000426          BR      6$          ;;GET OUT
21355 136416 005300 1$: DEC      RO          ;;ADJUST THE INDEX SO THAT IT WILL
21356 136420 006300          ASL     RO          ;; WORK FOR THE ERROR TABLE
21357 136422 006300          ASL     RO
21358 136424 006300          ASL     RO
21359 136426 062700 001324  ADD     @#$ERRTB,RO  ;;FORM TABLE POINTER
21360 136432 012037 136442  MOV     (RO)+,2$     ;;PICKUP "ERROR MESSAGE" POINTER
21361 136436 001404          BEQ     3$          ;;SKIP TYPEOUT IF NO POINTER
21362 136440 104401          TYPE          ;;
21363 136442 000000 2$: .WORD  0          ;;ERROR MESSAGE POINTER GOES HERE
21364 136444 104401 001175  TYPE     .$CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
21365 136450 012037 136460 3$: MOV     (RO)+,4$     ;;PICKUP "DATA HEADER" POINTER
21366 136454 001404          BEQ     5$          ;;SKIP TYPEOUT IF 0
21367 136456 104401          TYPE          ;;TYPE THE "DATA HEADER"
21368 136460 000000 4$: .WORD  0          ;; "DATA HEADER" POINTER GOES HERE
21369 136462 104401 001175  TYPE     .$CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
21370 136466 011000 5$: MOV     (RO),RO      ;;PICKUP "DATA TABLE" POINTER
21371 136470 001004          BNE     7$          ;;GO TYPE THE DATA
21372 136472 012600 6$: MOV     (SP)+,RO     ;;RESTORE RO
21373 136474 104401 001175  TYPE     .$CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
21374 136500 000207          RTS      PC          ;;RETURN
21375 136502 7$:
21376 136502 021027 000005  CMP     (RO),#5      ;;GENERAL PURPOSE REGISTER?
21377 136506 101021          BHI     9$          ;;IF NOT, GO TYPE DATA
21378 136510 042737 000700 136544  BIC     @BIT8:BIT7:BIT6,8$ ;;CLEAR BITS FOR SOURCE REGISTER
21379 136516 011037 001160          MOV     (RO),$TMPO  ;;SAVE (RO)
21380 136522 000337 001160          SWAB   $TMPO        ;;GET REGISTER NUMBER TO HIGH BYTE
21381 136526 006237 001160          ASR    $TMPO        ;;GET REGISTER NUMBER TO BITS 8-6
21382 136532 006237 001160          ASR    $TMPO
21383 136536 053737 001160 136544  BIS     $TMPO,8$    ;;SET BITS IN MOV INSTRUCTION
21384                                ;;ACCORDING TO REGISTER NUMBER
21385 136544 010046 8$: MOV     RO,-(SP)     ;;MOVE CONTEXT OF REGISTER TO STACK
21386 136546 005720          TST    (RO)+       ;;ADVANCE POINTER
21387 136550 000401          BR     10$         ;;GO TYPE

```

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

21388 136552 013046          9$:   MOV      @ (RO)+, -(SP)   ;; IF NOT GPR, SAVE @ (RO)+ FOR TYPEOUT
21389 136554 104402          10$:  TYPOC                      ;; GO TYPE--OCTAL ASCII (ALL DIGITS)
21390 136556 005710                      TST      (RO)                ;; IS THERE ANOTHER NUMBER?
21391 136560 001744                      BEQ      6$                  ;; BR IF NO
21392 136562 104401 136570          TYPE     ,11$                ;; TYPE TWO(2) SPACES
21393 136566 000745                      BR       7$
21394 136570 040 040 000 11$:  .ASCIZ  / /                    ;; TWO(2) SPACES
21395
21396
21397          .SBTTL  GLOBAL SUBROUTINES SECTION
21398
21399          ;++
21400          ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
21401          ; THAT ARE USED IN MORE THAN ONE TEST.
21402          ; -
21403
21404          ;++
21405          ; FUNCTIONAL DESCRIPTION:
21406          ;   SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS
21407
21408
21409          ; INPUTS: NONE
21410
21411          ; OUTPUTS: NONE
21412
21413          ; SUBORDINATE ROUTINES USED: LOAD PARS
21414          ;                               LOAD PDRS
21415
21416          ; FUNCTIONAL SIDE EFFECTS: NONE
21417
21418          ; CALLING SEQUENCE:   JSR      PC,INITMM
21419
21420 136574 012701 172240  INITMM: MOV      #172240,R1          ;BASE ADDRESS OF SIPARS
21421 136600 004737 136736  JSR      PC, LDPARS
21422 136604 012701 172260  MOV      #172260,R1          ;BASE ADDRESS OF SDPARS
21423 136610 004737 136736  JSR      PC, LDPARS
21424 136614 012701 172340  MOV      #172340,R1          ;BASE ADDRESS OF KIPARS
21425 136620 004737 136736  JSR      PC, LDPARS
21426 136624 012701 172360  MOV      #172360,R1          ;BASE ADDRESS OF KDPARS
21427 136630 004737 136736  JSR      PC, LDPARS
21428 136634 012701 177640  MOV      #177640,R1          ;BASE ADDRESS OF UIPARS
21429 136640 004737 136736  JSR      PC, LDPARS
21430 136644 012701 177660  MOV      #177660,R1          ;BASE ADDRESS OF UDPARS
21431 136650 004737 136736  JSR      PC, LDPARS
21432 136654 012701 177600  MOV      #177600,R1          ;BASE ADDRESS OF UIPDRS
21433 136660 004737 136766  JSR      PC, LDPDRS
21434 136664 012701 177620  MOV      #177620,R1          ;BASE ADDRESS OF UDPDRS
21435 136670 004737 136766  JSR      PC, LDPDRS
21436 136674 012701 172300  MOV      #172300,R1          ;BASE ADDRESS OF KIPDRS
21437 136700 004737 136766  JSR      PC, LDPDRS
21438 136704 012701 172320  MOV      #172320,R1          ;BASE ADDRESS OF KDPDRS
21439 136710 004737 136766  JSR      PC, LDPDRS
21440 136714 012701 172200  MOV      #172200,R1          ;BASE ADDRESS OF SIPDRS
21441 136720 004737 136766  JSR      PC, LDPDRS
21442 136724 012701 172220  MOV      #172220,R1          ;BASE ADDRESS OF SDPDRS
21443 136730 004737 136766  JSR      PC, LDPDRS
21444 136734 000207          RTS      PC                    ;RETURN

```

GLOBAL SUBROUTINES SECTION

```

21446
21447
21448
21449
21450
21451
21452
21453
21454
21455
21456
21457
21458
21459
21460
21461
21462
21463
21464
21465
21466 136736 012702 000006
21467 136742 005003
21468 136744 010321
21469 136746 062703 000200
21470 136752 077204
21471 136754 012721 002000
21472 136760 012711 177600
21473 136764 000207

```

```

; **
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS).
;   THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
;   SUPPLIED BY THE CALLING ROUTINE. PARS 0-5 WILL BE MAPPED FROM
;   ADDRESS 0 TO ADDRESS 137777 (0-24K). PAR 6 WILL BE MAPPED FROM
;   ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
;   PAGE.
;
; INPUTS:
;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE:      JSR      PC,LDPARS
LDPARS: MOV      #6,      R2          ;LET LOOP COUNTER COUNT FIRST 6 PARS
        CLR      R3          ;INITIALIZE INDEX VALUE
1$:     MOV      R3,      (R1)+     ;LOAD PARS
        ADD      #200,    R3       ;INDEX IN 4K INCREMENTS
        SOB     R2,      1$       ;LOAD FIRST SIX PARS
        MOV     #2000,   (R1)+     ;LET PAR6 MAP TO 200000
        MOV     #177600,(R1)     ;LET PAR7 MAP TO I/O PAGE
        RTS     PC             ;RETURN

```

GLOBAL SUBROUTINES SECTION

```

21475
21476
21477
21478
21479
21480
21481
21482
21483
21484
21485
21486
21487
21488
21489
21490
21491
21492
21493
21494
21495
21496 136766 012702 000006
21497 136772 012721 177406
21498 136776 077203
21499 137000 012721 077406
21500 137004 012711 077406
21501 137010 000207

```

```

; **
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DECRIPTOR REGISTERS (PDRS).
;   THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
;   SUPPLIED BY THE CALLING ROUTINE. PDRS 0-5 WILL BE INITIALIZED TO
;   4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
;   4K READ/WRITE NO BYPASS.
;   NOTE:  THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
;           THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
;
; INPUTS:
;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE:      JSR      PC,LDPARS
LDPDRS: MOV      #6,      R2           ;LET LOOP COUNTER COUNT FIRST 6 PARS
1$:    MOV      #177406,(R1)+       ;LOAD PDRS WITH 4K READ/WRITE BYPASS
      SOB      R2,      1$         ;LOAD FIRST SIX PDRS
      MOV      #77406,(R1)+       ;LET PAR6 BE 4K READ/WRITE NO BYPASS
      MOV      #77406,(R1)       ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
      RTS      PC                ;RETURN

```


GLOBAL SUBROUTINES SECTION

```

21503      ;**
21504      ; FUNCTIONAL DESCRIPTION:
21505      ;   SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
21506
21507      . INPUTS:
21508      ;   MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
21509
21510      ; OUTPUTS: NONE
21511
21512      ; SUBORDINATE ROUTINES USED: NONE
21513
21514      ; FUNCTIONAL SIDE EFFECTS: NONE
21515
21516      ; CALLING SEQUENCE: CALLED BY PARITY ABORT
21517      ;   MOV     @#114, SLOC00 ;SAVE CONTENTS OF PARITY ABORT VECTOR
21518      ;   MOV     @DSPAR, @#114 ;LET VECTOR POINT TO PARITY ABORT ROUTINE
21519
21520      ;
21521      ;   (CACHE PARITY ERROR OCCURS)
21522 137012 011637 001122      RAMPAR: MOV     (SP), $BADDR      ;STOR ADDRESS TRAPPED
21523 137016 032737 000100 177744      BIT     @BIT06, MSER      ;IF LOW BYTE PARITY ERROR
21524 137024 001401          BEQ     1$              ;THEN
21525 137026 104007          ERROR   +7              ;ERROR
21526 137030 032737 000200 177744 1$: BIT     @BIT07, MSER      ;IF HIGH BYTE PARITY ERROR
21527 137036 001401          BEQ     2$              ;THEN
21528 137040 104007          ERROR   +7              ;ERROR
21529 137042 032737 000040 177744 2$: BIT     @BIT05, MSER      ;IF TAG PARITY ERROR
21530 137050 001401          BEQ     3$              ;THEN
21531 137052 104007          ERROR   +7              ;ERROR
21532 137054 005037 177744      3$: CLR     MSER          ;INITIALIZE MSER AFTER ERROR
21533 137060 000002          RTI              ;RETURN
21534 137062 005237 002722      LKSINT: INC     LKSFL          ;INCREMENT FLAG
21535 137066 000002          RTI
21536
21537      .SBTTL Q22BE SIZE ROUTINE
21538      ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS. IF NONE
21539      ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES. THIS ROUTINE WILL
21540      ;ONLY RUN IN NOT UFD MODE.
21541
21542 137070 032737 001000 177750      Q22SIZ: BIT     @BIT09, MAIREG      ;UNIBUS SYSTEM?
21543 137076 001401          BEQ     1$              ;IF NOT, ADVANCE TO ROUTINE
21544 137100 0002C7          RTS     PC              ;OTHERWISE, RETURN
21545
21546      ;
21547      ; PREPARE TO DO SIZING
21548 137102 013701 000004      1$: MOV     ERRVEC, R1          ;STORE TIMEOUT VECTOR
21549 137106 012737 137262 000004      MOV     @7$, ERRVEC          ;POINT NEW TO PROGRAM
21550 137114 012737 000340 000006      MOV     @340, ERRVEC+2      ;AT PRIORITY 7
21551 137122 005037 001160          CLR     $TMP0              ;CLEAR Q22BE COUNTER
21552 137126 012702 170000          MOV     @170000, R2         ;FIRST POSSIBLE ADDRESS
21553 137132 012703 000510          MOV     @510, R3           ;VECTOR FOR IT
21554 137136 000404          BR     3$              ;TRY THOSE VALUES
21555
21556      ;
21557      ; NOW DO ACTUAL SIZING
21558 137140 062702 000020      2$: ADD     @20, R2          ;GET CSR FOR NEXT Q22BE
21559 137144 062703 000004          ADD     @4, R3            ;GET VECTOR FOR NEXT ONE

```

Q22BE SIZE ROUTINE

```

21560 137150 005712      3$:   TST      (R2)                ;TRY TO ACCESS CSR
21561                   ;
21562                   ; IF NO TIMEOUT, STORE EXISTING ADDRESSES TO REGISTERS
21563                   ;
21564 137152 005737 001160      TST      $TMP0                ;FIRST Q22BE FOUND?
21565 137156 001010      BNE      4$                    ;IF SECOND, BRANCH
21566 137160 012705 002664      MOV      @CSR1,R5             ;START WITH CSR1 FOR 1ST
21567 137164 010237 002702      MOV      R2,SIMGOA          ;SIMULTANEOUS GO
21568 137170 062737 000016 002702  ADD      @16,SIMGOA         ;ADDRESS
21569 137176 000402      BR       5$                    ;BRANCH TO INITIALISE
21570 137200 012705 002704      4$:   MOV      @CSR12,R5        ;START WITH CSR12 FOR 2ND
21571 137204 012704 000004      5$:   MOV      @4,R4           ;INITIALISE 5 REGISTERS
21572 137210 010215      MOV      R2,(R5)             ;INITIALISE CSR1
21573 137212 011565 000002      6$:   MOV      (R5),2(R5)      ;STORE TO NEXT ONE
21574 137216 005725      TST      (R5)+              ;GET NEXT ADDRESS
21575 137220 062715 000002      ADD      @2,(R5)            ;GET ADDRESS, POINT NEXT
21576 137224 077406      SOB      R4,6$              ;DO FOR NEXT 4 REGISTERS
21577 137226 010365 000002      MOV      R3,2(R5)          ;STORE INTERRUPT VECTOR
21578 137232 010365 000004      MOV      R3,4(R5)          ;AND PRIORITY
21579 137236 062765 000002 000004  ADD      @2,4(R5)
21580 137244 005237 001160      INC      $TMP0              ;COUNT Q22BE'S
21581 137250 022737 000002 001160  CMP      @2,$TMP0           ;TWO FOUND?
21582 137256 001406      BEQ      9$                    ;IF SO, STOP SIZING
21583 137260 000402      BR       8$                    ;OTHERWISE, CONTINUE SIZING
21584                   ;
21585                   ; ON TIMEOUT TRY TO LOOK AT NEXT ADDRESS RANGE
21586                   ;
21587 137262 005726      7$:   TST      (SP)+          ;RESTORE STACK FROM
21588 137264 005726      TST      (SP)+          ;TIMEOUT
21589 137266 022702 170160      8$:   CMP      @170160,R2    ;AT THE LAST POSSIBLE?
21590 137272 001322      BNE      2$                    ;IF NOT, BRANCH
21591 137274 005737 002664      9$:   TST      CSR1           ;1 FOUND?
21592 137300 001402      BEQ      10$                 ;IF NONE, BRANCH
21593 137302 104401 137314      TYPE    ,ONQ22              ;TYPE FOUND
21594 137306 010137 000004      10$:  MOV      R1,ERRVEC        ;RESTORE TIMEOUT VECTOR
21595 137312 000207      RTS     PC                   ;RETURN
21596
21597 137314      012      015      121  ONOQ22: .ASCIZ <12><15>/Q22BE USED DURING TESTING/
      137317      062      062      102
      137322      105      040      125
      137325      123      105      104
      137330      040      104      125
      137333      122      111      116
      137336      107      040      124
      137341      105      123      124
      137344      111      116      107
      137347      000
21598 .EVEN
21599 .SBTTL Q22BE INTERRUPT INITIALISE ROUTINE
21600 ;THIS ROUTINE WILL INITIALISE Q22BE TO INTERRUPT AT A PRIORITY AT (R0)+
21601 ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
21602 ;BY CLEARING GO.
21603
21604 137350 005013      Q22INT: CLR      (R3)                ;CLEAR TRANSFER TYPE IN CSR1
21605 137352 052710 000001      BIS      @BIT00,(R0)         ;ZERO DONE
21606 137356 011063 000002      MOV      (R0),2(R3)         ;SET PRIORITY IN CSR2
21607 137362 042710 000001      BIC      @BIT00,(R0)         ;PREPARE TO SET DONE

```

Q22BE INTERRUPT INITIALISE ROUTINE

```

21608 137366 000207          RTS      PC
21609
21610          .SBTTL DMATRN DATO CYCLE THRU Q22BE
21611          ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
21612          ;FOUND Q22BE STARTING AT LOCATION @CSR1. RO HAS 0 IF ONLY 1 TRANSFER IS
21613          ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
21614          ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
21615
21616 137370 012777 012525 043276 DMATRN: MOV      @12525,@DATA          ;DATA USED
21617 137376 005700          TST      RO              ;DO 1 WORD?
21618 137400 001404          BEQ      1$             ;IF YES, BRANCH
21619 137402 012777 001001 043256 MOV      @BIT09!BIT00,@CSR2 ;BLOCK MODE, GO
21620 137410 000414          BR       2$             ;BRANCH TO DO IT
21621 137412 012777 001601 043244 1$:  MOV      @1601,@CSR1      ;RESET LATENCY COUNT,DATO
21622 137420 012777 002740 043242 MOV      @TEMP,@BA       ;LOAD DMA ADDRESS
21623 137426 012777 177777 043236 MOV      @177777,@WC    ;DO 1 WORD
21624 137434 012777 000001 043224 MOV      @BIT00,@CSR2   ;DO IT
21625 137442 105777 043220 2$:  TSTB    @CSR2          ;DMA DONE?
21626 137446 100375          BPL      2$             ;WAIT TILL DONE
21627 137450 000207          3$:  RTS      PC          ;RETURN FROM SUBROUTINE
21628
21629          .SBTTL DMARD DATI THRU Q22BE
21630          ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
21631          ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. RO IS ZERO FOR SINGLE TRANSFER
21632
21633 137452 012777 002740 043210 DMARD: MOV      @TEMP,@BA          ;LOAD DMA ADDRESS
21634 137460 005700          TST      RO              ;DO 1 WORD?
21635 137462 001412          BEQ      1$             ;IF YES, BRANCH
21636 137464 012777 001507 043172 MOV      @1507,@CSR1    ;16 DATIB
21637 137472 012777 177770 043172 MOV      @177770,@WC   ;DO 8 WORD
21638 137500 012777 001001 043160 MOV      @BIT09!BIT00,@CSR2 ;BLOCK MODE GO
21639 137506 000411          BR       2$             ;GO CHECK
21640 137510 012777 001407 043146 1$:  MOV      @1407,@CSR1      ;RESET LATENCY COUNT,DATI
21641          ;LOAD NEW DATA TO DATA R.
21642 137516 012777 177777 043146 MOV      @177777,@WC   ;DO 1 WORD
21643 137524 012777 000001 043134 MOV      @BIT00,@CSR2   ;DO IT
21644 137532 105777 043130 2$:  TSTB    @CSR2          ;DMA DONE?
21645 137536 100375          BPL      2$             ;WAIT TILL DONE
21646 137540 000207          3$:  RTS      PC          ;RETURN FROM SUBROUTINE
21647
21648
21649
21650 137542 011637 001122 TOUT:  MOV      (SP),@BDADR      ;STORE TRAPPED PC
21651 137546 104130          ERROR   +130          ;UNEXPECTED TRAP
21652 137550 000002          RTI
21653
21654
21655          ;
21656          ;MMU GLOBAL SUBROUTINES
21657          ;
21658          ;
21659          ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
21660          ;
21661 137552 010046 MMU:  MOV      RO,-(SP)      ;SAVE CONTENTS OF REGISTERS
21662 137554 010146          MOV      R1,-(SP)      ;
21663 137556 010246          MOV      R2,-(SP)      ;
21664 137560 012700 177600 MOV      @177600,RO     ;
21665 137564 004737 137652 JSR      PC,PDR         ;INIT I AND D USER PDR'S

```

DMARD DATI THRU Q228E

```

21666 137570 004737 137674 JSR PC,PAR ;INIT I USER PAR S
21667 137574 004737 137674 JSR PC,PAR ;INIT D USER PAR'S
21668 137600 012700 172200 MOV #172200,R0
21669 137604 004737 137652 JSR PC,PDR ;INIT I AND D SUP PDR'S
21670 137610 004737 137674 JSR PC,PAR ;INI, I SUP PAR'S
21671 137614 004737 137674 JSR PC,PAR ;INIT D SUP PAR'S
21672 137620 004737 137652 JSR PC,PDR ;INIT I AND D KER PDR'S
21673 137624 004737 137674 JSR PC,PAR ;INIT I KER PAR'S
21674 137630 004737 137674 JSR PC,PAR ;INIT D KER PAR'S
21675 137634 012737 000027 172516 MOV #27,#172516 ;INIT MMR3
21676 137642 012602 MOV (SP)+,R2 ;RESTORE REGISTERS
21677 137644 012601 MOV (SP)+,R1
21678 137646 012600 MOV (SP)+,R0
21679 137650 000207 RTS PC ;RETURN
21680
21681 ;ROUTINE TO INITIALIZE PDR'S
21682 ;
21683 137652 005002 PDR: CLR R2 ;INIT CNTR
21684 137654 012720 077406 PDR1: MOV #77406,(R0)+ ;INIT PDR
21685 137660 062702 000001 ADD #1,R2 ;INCREMENT CNTR
21686 137664 022702 000020 CMP #16,R2 ;ARE WE DONE?
21687 137670 001371 BNE PDR1 ;BRANCH IF NOT
21688 137672 000207 RTS PC ;RETURN
21689
21690 ;ROUTINE TO INITIALIZE PAR'S
21691 ;
21692 137674 005001 PAR: CLR R1 ;SETUP TO INIT PAR
21693 137676 010120 PAR1: MOV R1,(R0)+ ;INIT PAR
21694 137700 062701 000200 ADD #200,R1 ;GET READY FOR NEXT PAR
21695 137704 022701 001600 CMP #1600,R1 ;REACHED A PAR?
21696 137710 001372 BNE PAR1 ;BRANCH IF NOT
21697 137712 012720 177600 MOV #177600,(R0)+ ;INIT PAR?
21698 137716 000207 RTS PC ;RETURN
21699
21700 ;TIME OUT ROUTINE
21701 ;
21702 137720 005205 ADDTRP: INC R5 ;INCREMENT TIME OUT FLAG
21703 137722 000002 RTI ;RETURN
21704
21705 ;MMU TRAP ROUTINE
21706 ;
21707 137724 023727 003030 000001 MMUTRP: CMP FLAG,#1 ;ARE WE EXPECTING AN ABORT
21708 137732 001401 BEQ 1$ ;YES GO ON
21709 137734 104002 ERROR +2 ;NO GO TO ERROR
21710 137736 010046 1$: MOV RO,-(SP) ;SAVE CONTENTS OF REG 0
21711 137740 013700 177776 MOV #177776,RO ;SAVE A COPY OF PSW
21712 137744 072027 177764 ASH #-14,RO ;LOOK AT BITS<15:14>
21713 137750 020027 000002 CMP RO,#2 ;WAS PS<15:14>=10
21714 137754 001001 BNE OK ;NO GO ON
21715 137756 000411 BR NOTOK ;YES CHANGE BITS TO 00
21716 137760 013700 177776 OK: MOV #177776,RO ;SAVE A COPY OF PSW
21717 137764 072027 000002 ASH #2,RO ;LOOK AT BITS<13:12>
21718 137770 072027 177764 ASH #-14,RO
21719 137774 020027 000002 CMP RO,#2
21720 140000 001002 BNE OK1 ;WAS PS<13:12>=10
21721 140002 005066 000004 NOTOK: CLR 4(SP) ;NO GO ON
21722 140006 013737 177572 003042 OK1: MOV #177572,SAVMRO ;CLEAR ILLEGAL MODE FFROM OLD PSW
;SAVE A COPY OF MMRO

```

DMARD DATI THRU Q228E

21723	140014	013737	177574	003044	MOV	00177574,SAVMR1	;SAVE A COPY OF MMR1
21724	140022	013737	177576	003046	MOV	00177576,SAVMR2	;SAVE A COPY OF MMR2
21725	140030	005037	177572		CLR	00177572	;CLEAR ABORT BITS AND TURN MMU OFF
21726	140034	005037	003030		CLR	FLAG	;CLEAR MMU ABORT FLAG
21727	140040	012600			MOV	(SP)+,R0	;RESTORE ORIGINAL CONTENTS OF REG 0
21728	140042	000002			RTI		;RETURN

DMARD DATI THRU 0228E

```

21732 ;FPP COMMON SUBROUTINES
21733 140044 012600 WLDTRP: MOV (SP)+,R0 ;SAVE PC
21734 140046 012605 MOV (SP)+,R5 ;SAVE STATUS AND RESTORE STACK
21735 140050 104003 ERROR +3
21736 140052 000110 JMP (R0) ;GO BACK INLINE
21737 ;
21738 ;
21739 ;
21740 140054 000000 TRPFLG: .WORD 0
21741 140056 000207 ERRFP: RTS R7
21742 140060 000207 ERR: RTS R7
21743 ;
21744 ;
21745 ;
21746 ;
21747 ;
21748 ;SUBROUTINE DATA VERFICATION -
21749 ;
21750 ; CALLED BY JSR R7,DATVER
21751 ;
21752 ;INPUT: (R4)=EXPECTED DATA
21753 ; (R1)=RECEIVED DATA
21754 ;
21755 ;THIS ROUTINE VERIFIES THAT THE 4 CONSECITIVE WORDS STARTING WITH (R4) ARE
21756 ;EQUAL TO THE FOUR WORDS ADDRESSED BY (R1). THE CONTENTS OF R4, AND R1 ARE NOT
21757 ;DISTURBED.
21758 ;LOCATION "COUNT" , IF NOT EQUAL TO 0 SIGNIFIES DATA ERROR
21759 ;IF THE STATUS IS FLOATING MODE, THE LAST TWO BYTES OF RECEIEVED
21760 ;ARE SIMPLY CHECKED FOR ZEROS
21761 ;
21762 ;
21763 140062 010446 DATVFR: MOV R4,-(SP) ;SAVE R4
21764 140064 010146 MOV R1,-(SP) ;SAVE R1
21765 140066 012737 000003 003120 MOV #3,COUNT ;SET UP ITERATION COUNT
21766 140074 000137 140112 JMP DAT1 ;
21767 ;
21768 140100 010446 DATVER: MOV R4,-(SP) ;SAVE R4
21769 140102 010146 MOV R1,-(SP) ;SAVE R1
21770 140104 012737 000005 003120 MOV #5,COUNT ;SFT UP ITERATION COUNT
21771 140112 005337 003120 DAT1: DEC COUNT
21772 140116 001402 BEQ 2$ ;BRANCH IF DONE
21773 140120 022421 CMP (R4)+,(R1)+ ;
21774 140122 001773 BEQ DAT1 ;
21775 140124 012601 2$: MOV (SP)+,R1 ;RESTORE R1
21776 140126 012604 MOV (SP)+,R4 ;RESTORE R4
21777 140130 000207 RTS R7 ;GO BACK TO CALLING ROUTINE
21778 ;IF DATA ERROR, COUNT NE 0

```

DMARD DATI THRU Q22BE

```

21780
21781
21782
21783
21784
21785
21786
21787
21788
21789
21790
21791
21792
21793
21794
21795
21796
21797
21798
21799
21800
21801
21802
21803 140132 032737 000400 177750
21804 140140 001007
21805 140142 000240
21806 140144 032737 000400 177750
21807 140152 001405
21808 140154 000240
21809 140156 000000
21810
21811 140160 104401 140174
21812 140164 000402
21813 140166 104401 140257
21814 140172 000207
21815
21816 140174 105 122 122
      140177 117 122 040
      140202 104 105 124
      140205 105 103 124
      140210 105 104 040
      140213 111 116 040
      140216 106 114 117
      140221 101 124 111
      140224 116 107 040
      140227 120 117 111
      140232 116 124 040
      140235 101 103 103
      140240 105 114 105
      140243 122 101 124
      140246 117 122 040
      140251 103 110 111
      140254 120 056 000
21817
21818 140257 105 122 122
      140262 117 122 040
      140265 104 105 124

```

;
 ;\$\$\$
 ; SUBROUTINE - DETERMINE FLOATING POINT ACCELERATOR (DETFPA)
 ;
 ; THIS SUBROUTINE IS CALLED IF AN ERROR IS DETECTED DURING EXECUTION OF THE
 ; FLOATING POINT TESTS.
 ; IT DETERMINES WHEATHER OR NOT THE FLOATING POINT ACCELERATOR CHIP OPTION
 ; IS PRESENT ON THE CPU BOARD AND PRINTS THE APPROPRIATE ERROR MESSAGE.
 ; THIS DETERMINATION IS MADE BASED ON THE "FPA AVAILABLE" FLAG, BIT 8
 ; OF THE MAINTENANCE REGISTER AT LOCATION 1777750. IF THE FPA BIT IS SET
 ; THEN THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED ON THE CPU BOARD AND
 ; AN ERROR MESSAGE IS PRINTED WHICH STATES THAT THE FLOATING POINT ERROR IS
 ; DUE TO THIS CHIP. OTHERWISE, THE J11 IS BLAMED FOR THE FLOATING POINT ERROR.
 ;
 ;\$\$\$
 ; CALLED BY: CALL @DETFPA ;\$\$\$
 ;
 ; INPUTS: NONE ;\$\$\$
 ;
 ; OUTPUTS: ERROR MESSAGES ;\$\$\$
 ;
 ; DETFPA: BIT @400,@MAIREG ;IS THE FPA HERE? \$\$\$
 BNE FPAOPT ;YES, BRANCH FPAOPT \$\$\$
 NOP ;DEBUG AID. \$\$\$
 ; 21806 140144 032737 000400 177750 BIT @400,@MAIREG ;IF NOT, \$\$\$
 BEQ NOFPA ;BRANCH TO NOFPA \$\$\$
 NOP ;DEBUG AID. \$\$\$
 HALT ; \$\$\$
 ;
 ; FPAOPT: TYPE .FPAFLT ; \$\$\$
 BR EXTFPA ; \$\$\$
 ; 21813 140166 104401 140257 NOFPA: TYPE .J11FLT ; \$\$\$
 EXTFPA: RTS PC ; \$\$\$
 ;
 ; FPAFLT: .ASCIZ /ERROR DETECTED IN FLOATING POINT ACCELERATOR CHIP./ ; \$\$\$
 ;
 ; J11FLT: .ASCIZ /ERROR DETECTED IN J11 FLOATING POINT PROCESSOR./ ; \$\$\$

DMARD DATI THRU Q22BE

140270	105	103	124
140273	105	104	040
140276	111	116	040
140301	112	061	061
140304	040	106	114
140307	117	101	124
140312	111	116	107
140315	040	120	117
140320	111	116	124
140323	040	120	122
140326	117	103	105
140331	123	123	117
140334	122	056	000

21819
21820

.EVEN

: \$\$\$

DMARD DATI THRU Q228E

21823
21824
21825

```

.SBTTL END OF PASS ROUTINE
;*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO LOOP
$EOP:
140340      140340 032757 000100 000052      BIT #BIT06,#52
140340      140340 001030                BNE $GET42
140350      140350 004537 126756                jsr r5,vireop
140354      140354 005037 001102                CLR $TSTNM          ;;ZERO THE TEST NUMBER
140360      140360 005037 001164                CLR $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
140364      140364 005237 001206                INC $PASS           ;;INCREMENT THE PASS NUMBER
140370      140370 042737 100000 001206        BIC #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
140376      140376 005327                DEC (PC)+           ;;LOOP?
140400      140400 000001                $EOPCT: .WORD 1
140402      140402 003022                BGT $DOAGN          ;;YES
140404      140404 012737                MOV (PC)+,#(PC)+   ;;RESTORE COUNTER
140406      140406 000001                $ENDCT: .WORD 1
140410      140410 140400                $EOPCT
140412      140412 104401 140457                TYPE , $ENDMG       ;;TYPE "END PASS #"
140416      140416 013746 001206                MOV $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
140422      140422 104405                TYPDS              ;;GO TYPE--DECIMAL ASCII WITH SIGN
140424      140424 104401 140454                TYPE , $ENULL       ;;TYPE A NULL CHARACTER
140430      140430 013700 000042        $GET42: MOV #42,R0   ;;GET MONITOR ADDRESS
140434      140434 001405                BEQ $DOAGN          ;;BRANCH IF NO MONITOR
140436      140436 000005                RESET              ;;CLEAR THE WORLD
140440      140440 004710                $LNDAD: JSR PC,(R0) ;;GO TO MONITOR
140442      140442 000240                NOP                ;;SAVE ROOM
140444      140444 000240                NOP                ;;FOR
140446      140446 000240                NOP                ;;ACT11
140450      140450                $DOAGN:
140450      140450 000137                JMP #(PC)+         ;;RETURN
140452      140452 004740                $RTNAD: .WORD LOOP
140454      140454 377 377 000                $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
140457      140457 015 012 105                $ENDMG: .ASCIZ <15><12>/END PASS #/
140462      140462 116 104 040
140465      140465 120 101 123
140470      140470 123 040 043
140473      140473 000

```

21826

```

.SBTTL SCOPE HANDLER ROUTINE
;*****
;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14=1 LOOP ON TEST
;*SW11=1 INHIBIT ITERATIONS
;*SW09=1 LOOP ON ERROR
;*SW08=1 LOOP ON TEST IN SWR<5:0>
;*CALL
;* SCOPE ;;SCOPE=IOT
140474      140474 104407                $SCOPE: CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR

```

SCOPE HANDLER ROUTINE

```

140476 052737 001000 177520      BIS #1000,BCSR ;ENABLE
140504 032777 040000 040426 1$:  BIT    #BIT14,#SWR      ;;LOOP ON PRESENT TEST?
140512 001117                      BNE    $OVER          ;;YES IF SW14=1
                                ;*****START OF CODE FOR THE XOR TESTER*****
140514 000416      $XTSTR: BR    6$      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
140516 013746 000004                      MOV    @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
140522 012737 140542 000004      MOV    #5,@#ERRVEC   ;;SET FOR TIMEOUT
140530 005737 177060                      TST    @#177060      ;;TIME OUT ON XOR?
140534 012637 000004      MOV    (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
140540 000466                      BR     $SVLAD        ;;GO TO THE NEXT TEST
140542 022626      5$:  CMP    (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
140544 012637 000004      MOV    (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
140550 000426                      BR     7$           ;;LOOP ON THE PRESENT TEST
140552                      6$:;*****END OF CODE FOR THE XOR TESTER*****
140552 032777 000400 040360      BIT    #BIT08,#SWR   ;;LOOP ON SPEC. TEST?
140560 001407                      BEQ    2$           ;;BR IF NO
140562 017746 040352      MOV    #SWR,-(SP)   ;;SET DESIRED TEST NUM. FROM SWR
140566 042716 000300      BIC    #SWRMK,(SP)  ;;STRIP AWAY UNDESIRED BITS
140572 122637 001102      CMPB   (SP)+,$TSTNM ;;ON THE RIGHT TEST?
140576 001465                      BEQ    $OVER        ;;BR IF YES
140600 105737 001103      2$:  TSTB   $ERFLG     ;;HAS AN ERROR OCCURRED?
140604 001421                      BEQ    3$           ;;BR IF NO
140606 123737 001115 001103      CMPB   $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
140614 101015                      BHI    3$           ;;BR IF NO
140616 032777 001000 040314      BIT    #BIT09,#SWR  ;;LOOP ON ERROR?
140624 001404                      BEQ    4$           ;;BR IF NO
140626 013737 001110 001106 7$:  MOV    $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
140634 000446                      BR     $OVER
140636 105037 001103      4$:  CLRB   $ERFLG     ;;ZERO THE ERROR FLAG
140642 005037 001164      CLR    $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
140646 000415                      BR     1$           ;;ESCAPE TO THE NEXT TEST
140650 032777 004000 040262 3$:  BIT    #BIT11,#SWR  ;;INHIBIT ITERATIONS?
140656 001011                      BNE    1$           ;;BR IF YES
140660 005737 001206      TST    $PASS       ;;IF FIRST PASS OF PROGRAM
140664 001406                      BEQ    1$           ;;      INHIBIT ITERATIONS
140666 005237 001104      INC    $ICNT       ;;INCREMENT ITERATION COUNT
140672 023737 001164 001104      CMP    $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
140700 002024                      BGE    $OVER        ;;BR IF MORE ITERATION REQUIRED
140702 012737 000001 001104 1$:  MOV    #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
140710 013737 140774 001164      MOV    $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
140716 105237 001102      $SVLAD: INCB   $TSTNM ;;COUNT TEST NUMBERS
140722 113737 001102 001204      MOVB   $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
140730 011637 001106      MOV    (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
140734 011637 001110      MOV    (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
140740 005037 001166      CLR    $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
140744 112737 000001 001115      MOVB   #1,$ERMAX   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
140752 013777 001102 040162 $OVER: MOV    $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
140760 013716 001106      MOV    $LPADR,(SP) ;;FUDGE RETURN ADDRESS
140764 042737 001000 177520      BIC    #1000,BCSR ;DISABLE
140772 000002                      RTI
140774 000001      $MXCNT: 1          ;;MAX. NUMBER OF ITERATIONS

```

21827

```

.SBTTL ERROR HANDLER ROUTINE
;*****
;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO ERTYPE ON ERROR

```

ERROR HANDLER ROUTINE

```

; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; *SW15=1      HALT ON ERROR
; *SW13=1      INHIBIT ERROR TYPEOUTS
; *SW10=1      BELL ON ERROR
; *SW09=1      LOOP ON ERROR
; *CALL
; *          ERROR  +N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
$ERROR:
140776      140776 005737 004120      TST      UQUIET      ;; TEST FOR USER-QUIET MODE
141002      141002 001403              BEQ      9$          ;; BRANCH IF FIELD-SERVICE MODE
141004      141004 005000              CLR      RO          ;; IN CASE RO HAS A #3 IN IT (+C)
141006      141006 004737 141220      JSR      PC,ABORT    ;; TEST FOR ABORT CONDITION
141012      141012 104407              9$:
141014      141014 052737 001000 177520      BIS      #1000,BCSR  ;; TEST FOR CHANGE IN SOFT SWR
;                               ;; ENABLE HALT ON BREAK
141022      141022 105237 001103      7$:      INCB      $ERFLG    ;; SET THE ERROR FLAG
141026      141026 001775              BEQ      7$          ;; DON'T LET THE FLAG GO TO ZERO
141030      141030 013777 001102 040104      MOV      $TSTNM,$DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
141036      141036 032777 002000 040074      BIT      #BIT10,$SWR  ;; BELL ON ERROR?
141044      141044 001402              BEQ      1$          ;; NO - SKIP
141046      141046 104401 001170              TYPE     , $BELL     ;; RING BELL
141052      141052 005237 001112      1$:      INC      $ERTTL     ;; COUNT THE NUMBER OF ERRORS
141056      141056 011637 001116      MOV      (SP), $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
141062      141062 162737 000002 001116      SUB      #2, $ERRPC
141070      141070 117737 040022 001114      MOV      # $ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
141076      141076 032777 020000 040034      BIT      #BIT13,$SWR  ;; SKIP TYPEOUT IF SET
141104      141104 001004              BNE     20$         ;; SKIP TYPEOUTS
141106      141106 004737 136356      JSR      PC,ERTYPE   ;; GO TO USER ERROR ROUTINE
141112      141112 104401 001175              TYPE     , $CRLF
141116      141116 122737 000001 001220      20$:     CHPB     #APTENV,$ENV  ;; RUNNING IN APT MODE
141124      141124 001007              BNE     2$          ;; NO, SKIP APT ERROR REPORT
141126      141126 113737 001114 141140      MOV      $ITEMB, 21$ ;; SET ITEM NUMBER AS ERROR NUMBER?
141134      141134 004737 141376      JSR      PC, $ATY4   ;; REPORT FATAL ERROR TO APT
141140      141140      000              21$:     .BYTE   0
141141      141141      000              .BYTE   0
141142      141142 000777              22$:     BR      22$         ;; APT ERROR LOOP
141144      141144 005777 037770      2$:      TST      $SWR      ;; HALT ON ERROR
141150      141150 100002              BPL     3$          ;; SKIP IF CONTINUE
141152      141152 000000              HALT    ;; HALT ON ERROR!
141154      141154 104407              CKSWR   ;; TEST FOR CHANGE IN SOFT-SWR
141156      141156 032777 001000 037754      3$:      BIT      #BIT09,$SWR  ;; LOOP ON ERROR SWITCH SET?
141164      141164 001402              BEQ     4$          ;; BR IF NO
141166      141166 013716 001110              MOV     $LPERR,(SP)  ;; FUDGE RETURN FOR LOOPING
141172      141172 005737 001165      4$:      TST      $ESCAPE    ;; CHECK FOR AN ESCAPE ADDRESS
141176      141176 001402              BEQ     5$          ;; BR IF NONE
141200      141200 013716 001166              MOV     $ESCAPE,(SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
141204      141204 022737 140440 000042      5$:      CMP      # $ENDAD, #42 ;; ACT-1' AUTO-ACCEPT?
141212      141212 001001              BNE     6$          ;; BRANCH IF NO
141214      141214 000000              HALT    ;; YES
141216      141216 000002              6$:      RTI      ;; RETURN
; *
; *SBTTL ABORT ROUTINE FOR LCP/ORION UFD MODE
141220      141220 005737 004116      ABORT:   TST      UDFLG      ;; TEST FOR USER FRIENDLY MODE
141224      141224 001454              BEQ     NOABRT      ;; IF NOT UFD THEN CONTINUE NORMAL OPERATION
141226      141226 020027 000032      CMP      RO,#32     ;; IS IT A +Z ?

```

ABORT ROUTINE FOR LCP/ORION UFD MODE

```

141232 001443      BEQ   ABORTZ      ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
141234 020027 000003  CMP   RO,#3        ;IS IS A ↑C ?
141240 001404      BEQ   ABORTC      ;BR TO LOAD ↑C ON XXDP+ STACK (NO ERROR)
141242 005737 004120  TST   UQU1ET      ;TEST FOR USER-QUIET MODE
141246 001443      BEQ   NOABRT      ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                     ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
141250 000422      BR    ABORTF      ;SET DRERR THEN LEAVE
141252 013737 004112 000030 ABORTC: MOV   SAV3C  ✓ ;RESTORE EMT LOCATION (30)
141260 013737 004114 000032  MOV   SAV32,32    ;RESTORE EMT PRIORITY LOCATION (32)
141266 104043      EMT   +43         ;GET XXDP STACK LOC. INTO RO FROM MONITOR
141270 005720      1$:  TST   (RO)+   ;FIND END OF STACK
141272 001376      BNE   1$
141274 112760 000057 177777  MOVB  #' /,-1(RO)  ;LOAD SLASH OVER ZERO
141302 112720 000136  MOVB  #' ↑.(RO)+   ;LOAD UPARROW
141306 112720 000103  MOVB  #' C.(RO)+   ;LOAD C
141312 105010      CLRB  (RO)        ;MAKE NEW END TO STACK
141314 000412      BR    ABORTZ      ;NOW LEAVE
141316 013737 004112 000030 ABORTE: MOV   SAV30,30 ;RESTORE EMT LOCATION (30)
141320 013737 004114 000032  MOV   SAV32,32    ;RESTORE EMT PRIORITY LOCATION (32)
141322 104042      EMT   +42         ;GET DCA LOCATION INTO RO FROM MONITOR
141334 012760 177777 000042  MOV   #-1,42(RO)  ;SET A -1 INTO LOCATION DRERR IN MONITOR
141342 013700 000042  ABORTZ: MOV   @42,RO ;AND PUT THE MONITOR RETURN ADDRESS IN RO
141346 005037 000042  CLR   @42         ;CLEAR MONITOR RETURN FLAG
141352 000137 140440  JMP   $ENDAD      ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
141356 000207      NOABRT: RTS   PC   ;IF NOTUFD RETURN TO MAINLINE

```

21828

```

.SBTTL APT COMMUNICATIONS ROUTINE
;*****
141360 112737 000001 141624 $ATY1: MOVB  #1,$FFLG ;:TO REPORT FATAL ERROR
141366 112737 000001 141622 $ATY3: MOVB  #1,$MFLG ;:TO TYPE A MESSAGE
141374 000403      BR    $ATYC
141376 112737 000001 141624 $ATY4: MOVB  #1,$FFLG ;:TO ONLY REPORT FATAL ERROR
141404 $ATYC:
141404 010046      MOV   RO,-(SP)    ;:PUSH RO ON STACK
141406 010146      MOV   R1,-(SP)    ;:PUSH R1 ON STACK
141410 105737 141622  TSTB  $MFLG      ;:SHOULD TYPE A MESSAGE?
141414 001450      BEQ   5$         ;:IF NOT: BR
141416 122737 000001 001220  CMPB  @APTENV,$ENV ;:OPERATING UNDER APT?
141424 001031      BNE   3$         ;:IF NOT: BR
141426 132737 000100 001221  BITB  @APTSPOOL,$ENVM ;:SHOULD SPOOL MESSAGES?
141434 001425      BEQ   3$         ;:IF NOT: BR
141436 017600 000004      MOV   @4(SP),RO   ;:GET MESSAGE ADDR.
141442 062766 000002 000004  ADD   @2,4(SP)    ;:BUMP RETURN ADDR.
141450 005737 001200      1$:  TST   $MSGTYPE  ;:SEE IF DONE W/ LAST XMISSION?
141454 001375      BNE   1$         ;:IF NOT: WAIT
141456 010037 001214      MOV   RO,$MSGAD   ;:PUT ADDR IN MAILBOX
141462 105720      2$:  TSTB  (RO)+   ;:FIND END OF MESSAGE
141464 001376      BNE   2$
141466 163700 001214      SUB   $MSGAD,RO   ;:SUB START OF MESSAGE
141472 006200      ASR   RO         ;:GET MESSAGE LNGTH IN WORDS
141474 010037 001216      MOV   RO,$MSGLGT  ;:PUT LENGTH IN MAILBOX
141500 012737 000004 001200  MOV   @4,$MSGTYPE ;:TELL APT TO TAKE MSG.
141506 000413      BR    5$
141510 017637 000004 141534 3$:  MOV   @4(SP),4$   ;:PUT MSG ADDR IN JSR LINKAGE
141516 062766 000002 000004  ADD   @2,4(SP)    ;:BUMP RETURN ADDRESS
141524 013746 177776      MOV   177776,-(SP) ;:PUSH 177776 ON STACK
141530 004737 141626      JSR   PC,$TYPE   ;:CALL TYPE MACRO
141534 000000      4$:  .WORD 0

```

APT COMMUNICATIONS ROUTINE

```

141536      5$:
141536 105737 141624 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
141542 001416      BEQ 12$ ;; IF NOT: BR
141544 005737 001220 TST $ENV ;; RUNNING UNDER APT?
141550 001413      BEQ 12$ ;; IF NOT: BR
141552 005737 001200 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
141556 001375      BNE 11$ ;; IF NOT: WAIT
141560 017637 000004 001202 MOV $4(SP), $FATAL ;; GET ERROR #
141566 062766 000002 000004 ADD #2, 4(SP) ;; BUMP RETURN ADDR.
141574 005237 001200 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
141600 105037 141624 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
141604 105037 141623 CLRB $LFLG ;; CLEAR LOG FLAG
141610 105037 141622 CLRB $MFLG ;; CLEAR MESSAGE FLAG
141614 012601      MOV (SP)+, R1 ;; POP STACK INTO R1
141616 012600      MOV (SP)+, R0 ;; POP STACK INTO R0
141620 000207      RTS PC ;; RETURN
141622 000      $MFLG: .BYTE 0 ;; MESSG. FLAG
141623 000      $LFLG: .BYTE 0 ;; LOG FLAG
141624 000      $FFLG: .BYTE 0 ;; FATAL FLAG
                .EVEN

```

000200
000001
000100
000040

21829

APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
.SBTTL TYPE ROUTINE

```

;*****
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
```

```

;*CALL:
;*1) USING A TRAP INSTRUCTION
;* TYPE .MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;* TYPE
;* MESADR
;*
```

```

141626 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
141632 100002      BPL 1$ ;; BR IF YES
141634 000000      HALT ;; HALT HERE IF NO TERMINAL
141636 000430      BR 3$ ;; LEAVE
141640 010046      1$: MOV RO, -(SP) ;; SAVE RO
141642 017600 000002 MOV $2(SP), R0 ;; GET ADDRESS OF ASCIZ STRING
141646 122737 000001 001220 CMPB $APTENV, $ENV ;; RUNNING IN APT MODE
141654 001011      BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
141656 132737 000100 001221 BITB $APTSPOOL, $ENVM ;; SPOOL MESSAGE TO APT
141664 001405      BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
141666 010037 141676 MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
141672 004737 141366 JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
141676 000000      61$: .WORD 0 ;; MESSAGE ADDRESS
141700 132737 000040 001221 62$: BITB $APTCSUP, $ENVM ;; APT CONSOLE SUPPRESSED
141706 001003      BNE 60$ ;; YES, SKIP TYP_ OUT
141710 112046      2$: MOVB (R0)+, (SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
141712 001005      BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
141714 005726      TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK

```

TYPE ROUTINE

```

141716 012600          60$:  MOV    (SP)+,R0      ;;RESTORE R0
141720 062716 000002  3$:  ADD    @2,(SP)      ;;ADJUST RETURN PC
141724 000002          RTI                    ;;RETURN
141726 122716 000011  4$:  CMPB   @HT,(SP)     ;;BRANCH IF <HT>
141732 001430          BEQ    @8$                    ;;
141734 122716 000200  CMPB   @CRLF,(SP)    ;;BRANCH IF NOT <CRLF>
141740 001006          BNE    @5$                    ;;
141742 005726          TST    (SP)+         ;;POP <CR><LF> EQUIV
141744 104401          TYPE                    ;;TYPE A CR AND LF
141746 001175          $CRLF
141750 105037 142156  CLRB   $CHARCNT     ;;CLEAR CHARACTER COUNT
141754 000755          BR    @2$           ;;GET NEXT CHARACTER
141756 004737 142040  5$:  JSR    PC,$TYPEC    ;;GO TYPE THIS CHARACTER
141752 123726 001156  6$:  CMPB   $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
141766 001350          BNE    @2$           ;;IF NO GO GET NEXT CHAR.
141770 013746 001154  MOV    $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
                               ;;AND THE NULL CHAR.
141774 105366 000001  7$:  DECB   1(SP)        ;;DOES A NULL NEED TO BE TYPED?
142000 002770          BLT    @6$           ;;BR IF NO--GO POP THE NULL OFF OF STACK
142002 004737 142040  JSR    PC,$TYPEC    ;;GO TYPE A NULL
142006 105337 142156  DECB   $CHARCNT     ;;DO NOT COUNT AS A COUNT
142012 000770          BR    @7$           ;;LOOP
;HORIZONTAL TAB PROCESSOR
142014 112716 000040  8$:  MOVB   @',(SP)     ;;REPLACE TAB WITH SPACE
142020 004737 142040  9$:  JSR    PC,$TYPEC    ;;TYPE A SPACE
142024 132737 000007 142156 BITB   @7,$CHARCNT  ;;BRANCH IF NOT AT
142032 001372          BNE    @9$           ;;TAB STOP
142034 005726          TST    (SP)+         ;;POP SPACE OFF STACK
142036 000724          BR    @2$           ;;GET NEXT CHARACTER
142040          $TYPEC:
142040 105777 037100  TSTB   @TKS         ;;CHAR IN KYBD BUFFER?
142044 100022          BPL    @10$         ;;BR IF NOT
142046 017746 037074  MOV    @TKB,-(SP)   ;;GET CHAR
142052 042716 177600  BIC    @177600,(SP) ;;STRIP EXTRANEIOUS BITS
142056 122716 000023  CMPB   @XOFF,(SP)  ;;WAS CHAR XOFF
142062 001012          BNE    @102$        ;;BR IF NOT
142064          101$:
142064 105777 037054  TSTB   @TKS         ;;WAIT FOR CHAR
142070 100375          BPL    @101$        ;;
142072 117716 037050  MOVB   @TKB,(SP)   ;;GET CHAR
142076 042716 177600  BIC    @177600,(SP) ;;STRIP IT
142102 122716 000021  CMPB   @XON,(SP)   ;;WAS IT XON?
142106 001366          BNE    @101$        ;;BR IF NOT
142110          102$:
142110 005726          TST    (SP)+         ;;FIX STACK
142112          10$:
142112 105777 037032  TSTB   @TPS         ;;WAIT UNTIL PRINTER IS READY
142116 100375          BPL    @10$         ;;
142120 116677 000002 037024 MOVB   2(SP),@TPB   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
142126 122766 000015 000002 CMPB   @CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
142134 001003          BNE    @1$           ;;BRANCH IF NO
142136 105037 142156  CLRB   $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
142142 000406          BR    $TYPEX        ;;EXIT
142144 122766 000012 000002 1$:  CMPB   @LF,2(SP)    ;;IS CHARACTER A LINE FEED?
142152 001402          BEQ    $TYPEX        ;;BRANCH IF YES
142154 105227          INCB  (PC)+       ;;COUNT THE CHARACTER
142156 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE

```

TYPE ROUTINE

21830 142160 000207

142162 017646 000000
 142166 116637 000001 142405
 142174 112637 142407
 142200 062716 000002
 142204 000406
 142206 112737 000001 142405
 142214 112737 000006 142407
 142222 112737 000005 142404
 142230 010346
 142232 010446
 142234 010546
 142236 113704 142407
 142242 005404
 142244 062704 000006
 142250 110437 142406
 142254 113704 142405
 142260 016605 000012
 142264 005003
 142266 006105
 142270 000404
 142272 006105
 142274 006105
 142276 006105
 142300 010503
 142302 006103
 142304 105337 142406
 142310 100016
 142312 042703 177770
 142316 001002
 142320 005704
 142322 001403
 142324 005204
 142326 052703 000060

```

$TYPEX: RTS      PC
.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;*OCTAL (ASCII) NUMBER AND TYPE IT.
;*$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
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;*$CALL:
;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
;*      TYPOC    ;;CALL FOR TYPEOUT
$TYPOS: MOV      @ (SP),-(SP)    ;;PICKUP THE MODE
        MOVVB   1(SP),%OFILL    ;;LOAD ZERO FILL SWITCH
        MOVVB   (SP)+,%OMODE+1  ;;NUMBER OF DIGITS TO TYPE
        ADD     #2,(SP)        ;;ADJUST RETURN ADDRESS
        BR      $TYPON
$TYPOC: MOVVB   #1,%OFILL      ;;SET THE ZERO FILL SWITCH
        MOVVB   #6,%OMODE+1    ;;SET FOR SIX(6) DIGITS
$TYPON: MOVVB   #5,%OCNT      ;;SET THE ITERATION COUNT
        MOV     R3,-(SP)       ;;SAVE R3
        MOV     R4,-(SP)       ;;SAVE R4
        MOV     R5,-(SP)       ;;SAVE R5
        MOVVB   %OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG     R4
        ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
        MOVVB   R4,%OMODE     ;;SAVE IT FOR USE
        MOVVB   %OFILL,R4     ;;GET THE ZERO FILL SWITCH
        MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
        CLR     R3            ;;CLEAR THE OUTPUT WORD
1$:     ROL     R5            ;;ROTATE MSB INTO "C"
        BR     3$            ;;GO DO MSB
2$:     ROL     R5            ;;FORM THIS DIGIT
        ROL     R5
        ROL     R5
        MOV     R5,R3
3$:     ROL     R3            ;;GET LSB OF THIS DIGIT
        DECB   %OMODE        ;;TYPE THIS DIGIT?
        BPL    7$            ;;BR IF NO
        BIC   #177770,R3     ;;GET RID OF JUNK
        BNE   4$            ;;TEST FOR 0
        TST   R4            ;;SUPPRESS THIS 0?
        BEQ   5$            ;;BR IF YES
4$:     INC    R4            ;;DON'T SUPPRESS ANYMORE 0'S
        BIS   #'0,R3        ;;MAKE THIS DIGIT ASCII
    
```

BINARY TO OCTAL (ASCII) AND TYPE

```

142332 052703 000040      5$:  BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
142336 110337 142402      MOVB     R3,R#      ;;SAVE FOR TYPING
142342 104401 142402      TYPE     ,#        ;;GO TYPE THIS DIGIT
142346 105337 142404      7$:  DECB     $OCNT   ;;COUNT BY 1
142352 003347      BGT      2$        ;;BR IF MORE TO DO
142354 002402      BLT      6$        ;;BR IF DONE
142356 005204      INC      R4        ;;INSURE LAST DIGIT ISN'T A BLANK
142360 000744      BR       2$        ;;GO DO THE LAST DIGIT
142362 012605      6$:  MOV      (SP)+,R5  ;;RESTORE R5
142364 012604      MOV      (SP)+,R4  ;;RESTORE R4
142366 012603      MOV      (SP)+,R3  ;;RESTORE R3
142370 016666 000002 000004  MOV      2(SP),4(SP) ;;SET THE STACK FOR RETURNING
142376 012616      MOV      (SP)+,(SP)
142400 000002      RTI                       ;;RETURN
142402 000      8$:  .BYTE    0      ;;STORAGE FOR ASCII DIGIT
142403 000      .BYTE    0      ;;TERMINATOR FOR TYPE ROUTINE
142404 000      $OCNT: .BYTE    0      ;;OCTAL DIGIT COUNTER
142405 000      $OFILL: .BYTE   0      ;;ZERO FILL SWITCH
142406 000000      $OMODE: .WORD   0      ;;NUMBER OF DIGITS TO TYPE
21831      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5 DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*CALL:
;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS                    ;;GO TO THE ROUTINE
;TYPDS:
142410      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
142410 010046      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
142412 010146      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
142414 010246      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
142416 010346      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
142420 010546      MOV      $20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
142422 012746 020200      MOV      20(SP),R5     ;;GET THE INPUT NUMBER
142426 016605 000020      BPL      1$          ;;BR IF INPUT IS POS.
142432 100004      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
142434 005405      MOVB     #' -,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
142436 112766 000055 000001  1$:  CLR      R0          ;;ZERO THE CONSTANTS INDEX
142444 005000      MOV      $DBLK,R3     ;;SETUP THE OUTPUT POINTER
142446 012703 142624      MOVB     #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
142452 112723 000040      2$:  CLR      R2          ;;CLEAR THE BCD NUMBER
142456 005002      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
14246 016001 142614      3$:  SUB      R1,R5      ;;FORM THIS BCD DIGIT
14247 160105      BLT      4$          ;;BR IF DONE
142470 002402      INC      R2          ;;INCREASE THE BCD DIGIT BY 1
142472 000774      BR       3$
142474 060105      4$:  ADD      R1,R5      ;;ADD BACK THE CONSTANT
142476 005702      TST      R2          ;;CHECK IF BCD DIGIT=0
142500 001002      BNE      5$          ;;FALL THROUGH IF 0
142502 105716      TSTB    (SP)         ;;STILL DOING LEADING 0'S?
142504 100407      BMI      7$          ;;BR IF YES
142506 106316      5$:  ASLB    (SP)         ;;MSD?
142510 103003      BCC      6$          ;;BR IF NO
142512 116663 000001 177777  MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN

```


CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

142520 052702 000060      6$:  BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
142524 052702 000040      7$:  BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
142530 110223              MOVVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
142532 005720              TST      (R0)+      ;;JUST INCREMENTING
142534 020027 000010      CMP      R0,#10    ;;CHECK THE TABLE INDEX
142540 002746              BLT      2$        ;;GO DO THE NEXT DIGIT
142542 003002              BGT      8$        ;;GO TO EXIT
142544 010502              MOV      R5,R2     ;;GET THE LSD
142546 000764              BR       6$        ;;GO CHANGE TO ASCII
142550 105726              8$:  TSTB   (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
142552 100003              BPL      9$        ;;BR IF NO
142554 116663 177777 177776 9$:  MOVVB   -(Sf),-2(R3) ;;YES--SET THE SIGN FOR TYPING
142562 105013              CLRB   (R3)       ;;SET THE TERMINATOR
142564 012605              MOV     (SP)+,R5   ;;POP STACK INTO R5
142566 012603              MOV     (SP)+,R3   ;;POP STACK INTO R3
142570 012602              MOV     (SP)+,R2   ;;POP STACK INTO R2
142572 012601              MOV     (SP)+,R1   ;;POP STACK INTO R1
142574 012600              MOV     (SP)+,R0   ;;POP STACK INTO R0
142576 104401 142624      TYPE    ,#DBLK     ;;NOW TYPE THE NUMBER
142602 016666 000002 000004 MOV     2(SP),4(SP) ;;ADJUST THE STACK
142610 012616              MOV     (SP)+,(SP)
142612 000002              RTI                    ;;RETURN TO USER
142614 023420      $DTBL: 10000.
142616 001750              1000.
142620 000144              100.
142622 000012              10.
142624              $DBLK: .BLKW 4

```

21832

```

.SBTTL TTY INPUT ROUTINE
;*****
.ENABL LSB
;*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.
142634 022737 000176 001140 $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
142642 001074              BNE     15$         ;;BRANCH IF NO
142644 105777 036274              TSTB   @TKS        ;;CHAR THERE?
142650 100071              BPL     15$         ;;IF NO, DON'T WAIT AROUND
142652 117746 036270              MOVVB  @TKB,-(SP)   ;;SAVE THE CHAR
142656 042716 177600              BIC    #C177,(SP)  ;;STRIP-OFF THE ASCII
142662 022726 000007              CMP     #7,(SP)+   ;;IS IT A CONTROL G?
142666 001062              BNE     15$         ;;NO, RETURN TO USER
142670 123727 001134 000001      CMPB   $AUTOB,#1   ;;ARE WE RUNNING IN AUTO-MODE?
142676 001456              BEQ     15$         ;;BRANCH IF YES
142700 104401 143371              TYPE   ,#CNTLG    ;;ECHO THE CONTROL G (+G)
142704 104401 143376      $GTSWR: TYPE   ,#MSWR  ;;TYPE CURRENT CONTENTS
142710 013746 000176              MOV    SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
142714 104402              TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
142716 104401 143407              TYPE   ,#MNEW     ;;PROMPT FOR NEW SWR
142722 005046              19$:  CLR     -(SP)   ;;CLEAR COUNTER
142724 005046              CLR     -(SP)     ;;THE NEW SWR
142726 105777 036212      7$:  TSTB   @TKS        ;;CHAR THERE?
142732 100375              BPL     7$         ;;IF NOT TRY AGAIN
142734 117746 036206              MOVVB  @TKB,-(SP) ;;PICK UP CHAR
142740 042716 177600              BIC    #C177,(SP) ;;MAKE IT 7-BIT ASCII
142744 021627 000025      9$:  CMP     (SP),#25  ;;IS IT A CONTROL-U?

```

TTY INPUT ROUTINE

```

142750 001005          BNE      10$          ;;BRANCH IF NOT
142752 104401 143364   TYPE     , $CNTRLU  ;;YES, ECHO CONTROL-U (+U)
142756 062706 000006   20$:    ADD      #6,SP      ;;IGNORE PREVIOUS INPUT
142762 000757          BR       19$          ;;LET'S TRY IT AGAIN
142764 021627 000015   10$:    CMP      (SP),#15    ;;IS IT A <CR>?
142770 001022          BNE      16$          ;;BRANCH IF NO
142772 005766 000004   TST     4(SP)        ;;YES, IS IT THE FIRST CHAR?
142776 001403          BEQ      11$          ;;BRANCH IF YES
143000 016677 000002 036132   MOV     2(SP),@SWR   ;;SAVE NEW SWR
143006 062706 000006   11$:    ADD      #6,SP      ;;CLEAR UP STACK
143012 104401 001175   14$:    TYPE     , $CRLF    ;;ECHO <CR> AND <LF>
143016 123727 001135 000001   CMPB   $INTAG,#1    ;;RF-ENABLE TTY KBD INTERRUPTS?
143024 001003          BNE      15$          ;;BRANCH IF NOT
143026 012777 000100 036110   MOV     #100,@TKS   ;;RE-ENABLE TTY KBD INTERRUPTS
143034 000002          15$:    RTI          ;;RETURN
143036 004737 142040   16$:    JSR     PC,$TYPEC  ;;ECHO CHAR
143042 021627 000060   CMP     (SP),#60    ;;CHAR < 0?
143046 002420          BLT     18$          ;;BRANCH IF YES
143050 021627 000067   CMP     (SP),#67    ;;CHAR > 7?
143054 003015          BGT     18$          ;;BRANCH IF YES
143056 042726 000060   BIC     #60,(SP)+   ;;STRIP-OFF ASCII
143062 005766 000002   TST     2(SP)        ;;IS THIS THE FIRST CHAR
143066 001403          BEQ     17$          ;;BRANCH IF YES
143070 006316          ASL     (SP)         ;;NO, SHIFT PRESENT
143072 006316          ASL     (SP)         ;; CHAR OVER TO MAKE
143074 006316          ASL     (SP)         ;; ROOM FOR NEW ONE.
143076 005266 000002   17$:    INC     2(SP)        ;;KEEP COUNT OF CHAR
143102 056616 177776   BIS     -2(SP),(SP) ;;SET IN NEW CHAR
143106 000707          BR      7$          ;;GET THE NEXT ONE
143110 104401 001174   18$:    TYPE     , $QUES   ;;TYPE ?<CR><LF>
143114 000720          BR      20$          ;;SIMULATE CONTROL-U
.DSABL  L$B
;*****
;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;*CALL:
;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
;*                  ;;WITH PARITY BIT STRIPPED OFF
;
143116 011646          $RDCHR: MOV     (SP),-(SP)  ;;PUSH DOWN THE PC
143120 016666 000004 000002   MOV     4(SP),2(SP) ;;SAVE THE PS
143126 105777 036012   1$:    TSTB   @TKS      ;;WAIT FOR
143132 100375          BPL     1$          ;;A CHARACTER
143134 117766 036006 000004   MOVB   @TKB,4(SP)   ;;READ THE TTY
143142 042766 177600 000004   BIC     #C<177>,4(SP) ;;GET RID OF JUNK IF ANY
143150 026627 000004 000023   CMP     4(SP),#23   ;;IS IT A CONTROL-S?
143156 001013          BNE     3$          ;;BRANCH IF NO
143160 105777 035760   2$:    TSTB   @TKS      ;;WAIT FOR A CHARACTER
143164 100375          BPL     2$          ;;LOOP UNTIL ITS THERE
143166 117746 035754   MOVB   @TKB,-(SP)   ;;GET CHARACTER
143172 042716 177600   BIC     #C177,(SP)  ;;MAKE IT 7-BIT ASCII
143176 022627 000021   CMP     (SP)+,#21   ;;IS IT A CONTROL-Q?
143202 001366          BNE     2$          ;;IF NOT DISCARD IT
143204 000750          BR      1$          ;;YES, RESUME
143206 026627 000004 000021 3$:    CMP     4(SP),#$XON  ;;IS IT A RANDOM XON?
143214 001744          BEQ     1$          ;;BRANCH IF YES
143216 026627 000004 000140   CMP     4(SP),#140  ;;IS IT UPPER CASE?
;RAN001
;RAN001

```

TTY INPUT ROUTINE

```

143224 002407          BLT      4$          ;;BRANCH IF YES
143226 026627 000004 000175    CMP      4(SP),0175    ;;IS IT A SPECIAL CHAR?
143234 003003          BGT      4$          ;;BRANCH IF YES
143236 042766 000040 000004    BIC      040,4(SP)    ;;MAKE IT UPPER CASE
143244 000002          4$:    RTI          ;;GO BACK TO USER
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;*      RDLIN          ;;INPUT A STRING FROM THE TTY
;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;*      TERMINATOR WILL BE A BYTE OF ALL 0'S
143246 010346          $RDLIN: MOV     R3,-(SP)    ;;SAVE R3
143250 012703 143354    1$:    MOV     0$TTYIN,R3    ;;GET ADDRESS
143254 022703 143364    2$:    CMP     0$TTYIN+8.,R3    ;;BUFFER FULL?
143260 101405          BLOS     4$          ;;BR IF YES
143262 104410          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
143264 112613          MOV     (SP)+,(R3)    ;;GET CHARACTER
143266 122713 000177    10$:   CMP     0177,(R3)    ;;IS IT A RUBOUT
143272 001003          BNE     3$          ;;SKIP IF NOT
143274 104401 001174    4$:    TYPE   ,0QUES    ;;TYPE A '?'
143300 000763          BR      1$          ;;CLEAR THE BUFFER AND LOOP
143302 111337 143352    3$:    MOV     (R3),9$    ;;ECHO THE CHARACTER
143306 104401 143352          TYPE   ,9$
143312 122723 000015          CMP     015,(R3)+    ;;CHECK FOR RETURN
143316 001356          BNE     2$          ;;LOOP IF NOT RETURN
143320 105063 177777          CLRB   -1(R3)    ;;CLEAR RETURN (THE 15)
143324 104401 001176          TYPE   ,0LF      ;;TYPE A LINE FEED
143330 012603          MOV     (SP)+,R3    ;;RESTORE R3
143332 011646          MOV     (SP),-(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE
143334 016666 000004 000002    MOV     4(SP),2(SP)  ;;
143342 012766 143354 000004    MOV     0$TTYIN,4(SP)
143350 000002          RTI          ;;RETURN
143352 000          9$:    .BYTE   0          ;;STORAGE FOR ASCII CHAR. TO TYPE
143353 000          .BYTE   0          ;;TERMINATOR
143354          $TTYIN: .BLKB  8.    ;;RESERVE 8 BYTES FOR TTY INPUT
143364 136 125 015    $CNTLU: .ASCIZ /+U/<15><12> ;;CONTROL "U"
143367 012 000          $CNTLG: .ASCIZ /+G/<15><12> ;;CONTROL "G"
143371 136 107 015    $MSWR: .ASCIZ <15><12>/SWR = /
143374 012 000          $MNEW: .ASCIZ / NEW = /
143376 015 012 123
143401 127 122 040
143404 075 040 000
143407 040 040 116
143412 105 127 040
143415 075 040 000

```

21833

```

.SBTTL READ AN OCTAL NUMBER FROM THE TTY
;*****
;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;CHANGE IT TO BINARY.
;CALL:
;*      RDOCT          ;;READ AN OCTAL NUMBER
;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
;*      HIGH ORDER BITS ARE IN $HIOCT
143420 011646          $RDOCT: MOV     (SP), (SP)    ;;PROVIDE SPACE FOR THE
143422 016666 000004 000002    MOV     4(SP),2(SP)  ;;INPUT NUMBER
143430 010046          MOV     R0,-(SP)    ;;PUSH R0 ON STACK
143432 010146          MOV     R1,-(SP)    ;;PUSH R1 ON STACK

```

READ AN OCTAL NUMBER FROM THE TTY

```

143434 010246          MOV     R2,-(SP)      ;;PUSH R2 ON STACK
143436 104411          RDLIN                    ;;READ AN ASCII LINE
143440 012600          MOV     (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
143442 005001          CLR     R1             ;;CLEAR DATA WORD
143444 005002          CLR     R2
143446 112046          2$:  MOVVB  (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
143450 001412          BEQ    3$             ;;IF ZERO GET OUT
143452 006301          ASL    R1             ;;*2
143454 006102          ROL    R2
143456 006301          ASL    R1             ;;*4
143460 006102          ROL    R2
143462 006301          ASL    R1             ;;*8
143464 006102          ROL    R2
143466 042716 177770  BIC    #+C7,(SP)     ;;STRIP THE ASCII JUNK
143472 062601          ADD    (SP)+,R1      ;;ADD IN THIS DIGIT
143474 000764          BR     2$            ;;LOOP
143476 005726          3$:  TST    (SP)+      ;;CLEAN TERMINATOR FROM STACK
143500 010166 000012  MOV    R1,12(SP)     ;;SAVE THE RESULT
143504 010237 143520  MOV    R2,$HIOCT
143510 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
143512 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
143514 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
143516 000002          RTI                    ;;RETURN
143520 000000          $HIOCT: .WORD 0      ;;HIGH ORDER BITS GC HERE
21834
.SBTTL TRAP DECODER
;*****
;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;GO TO THAT ROUTINE.
143522 010046          $TRAP: MOV    R0,-(SP)  ;;SAVE R0
143524 016600 000002  MOV    2(SP),R0     ;;GET TRAP ADDRESS
143530 005740          TST    -(R0)        ;;BACKUP BY 2
143532 111000          MOVVB  (R0),R0      ;;GET RIGHT BYTE OF TRAP
143534 006300          ASL    R0           ;;POSITION FOR INDEXING
143536 016000 143556  MOV    $TRPAD(R0),R0 ;;INDEX TO TABLE
143542 000200          RTS    R0           ;;GO TO ROUTINE
;THIS IS USE TO HANDLE THE "GETPRI" MACRO
143544 011646          $TRAP2: MOV   (SP)-,(SP) ;;MOVE THE PC DOWN
143546 016666 000004 000002 MOV   4(SP),2(SP)  ;;MOVE THE PSW DOWN
143554 000002          RTI                    ;;RESTORE THE PSW
.SBTTL TRAP TABLE
;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;BY THE "TRAP" INSTRUCTION.
;
; ROUTINE
;
;-----
143556 143544          $TRPAD: .WORD  $TRAP2
143560 141626          $TYPE   ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
143562 142206          $TYPOC  ;;CALL=TYPOC  TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
143564 142162          $TYPOS  ;;CALL=TYPOS  TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
143566 142222          $TYPON  ;;CALL=TYPON   TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
143570 142410          $TYPDS  ;;CALL=TYPDS  TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
143572 142704          $GTSWR  ;;CALL=GTSWR  TRAP+6(104406) GET SOFT-SWR SETTING
143574 142634          $CKSWR  ;;CALL=CKSWR  TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
143576 143116          $RDCHR  ;;CALL=RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
143600 143246          $RDLIN  ;;CALL=RDLIN  TRAP+11(104411) TTY TYPEIN STRING ROUTINE
143602 143420          $RDOCT  ;;CALL=RDOCT  TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

```

POWER DOWN AND UP ROUTINES

21835

```

.SBTTL POWER DOWN AND UP ROUTINES
;*****
;POWER DOWN ROUTINE
143604 012737 143744 000024 $PWRDN: MOV @ILLUP,@PWRVEC ;;SET FOR FAST UP
143612 012737 000340 000026 MOV @340,@PWRVEC+2 ;;PRIO:7
143620 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
143622 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
143624 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
143626 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
143630 010446 MOV R4,(SP) ;;PUSH R4 ON STACK
143632 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
143634 017746 035300 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
143640 010637 143750 MOV SP,$SAVR6 ;;SAVE SP
143644 012737 143656 000024 MOV @PWRUP,@PWRVEC ;;SET UP VECTOR
143652 000000 HALT
143654 000776 BR -2 ;;HANG UP
;*****
;POWER UP ROUTINE
143656 012737 143744 000024 $PWRUP: MOV @ILLUP,@PWRVEC ;;SET FOR FAST DOWN
143664 013706 143750 MOV $SAVR6,SP ;;GET SP
143670 005037 143750 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
143674 005237 143750 1$: INC $SAVR6 ;;WAIT FOR THE INC
143700 001375 BNE 1$ ;;OF WORD
143702 012677 035232 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
143706 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
143710 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
143712 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
143714 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
143716 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
143720 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
143722 012737 143604 000024 MOV @PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
143730 012737 000340 000026 MOV @340,@PWRVEC+2 ;;PRIO:7
143736 104401 TYPE ;;REPORT THE POWER FAILURE
143740 143752 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
143742 000002 RTI
143744 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
143746 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
143750 000000 $SAVR6: 0 ;;PUT THE SP HERE
143752 015 012 120 $POWER: .ASCIZ <15><12>"POWER"
143755 117 127 105
143760 122 000

.EVEN
.END

```

21837

000001

.EVEN
.END

Symbol table

ABASE = 000000	AMADR3 = 000000	BIT14 = 040000	DAT80 124574	D4 052760
ABOEXT 111314	AMADR4 = 000000	BIT15 = 100000	DATI 125124	D5 052774
ABORT 141220	AMAMS1 = 000000	BIT2 = 000004	DATVER 140100	D6 053004
ABORTC 141252	AMAMS2 = 000000	BIT3 = 000010	DATVFR 140062	D7 053010
ABORTE 141316	AMAMS3 = 000000	BIT4 = 000020	DAT1 140112	EPPAS 003034
ABORTI 107634	AMAMS4 = 000000	BIT5 = 000040	DCOUNT 114144	EMTOA 030464
ABORTR 107676	AMSGAD = 000000	BIT6 = 000100	DDIS = 177570	EMTOB 030472
ABORTZ 141342	AMSGLG = 000000	BIT7 = 000200	DELAY 116754	EMTSAV 004024
ABORTO 044016	AMSGTY = 000000	BIT8 = 000400	DETFPA 140132	EMTVEC = 000030
ABORT7 044204	AMTYP1 = 000000	BIT9 = 001000	DH1 134621	EM1 127002
ABROUT 111146	AMTYP2 = 000000	BNO 043746	DH105 135562	EM10 127254
ACDW1 = 000000	AMTYP3 = 000000	BN1 044136	DH115 135614	EM100 132570
ACDW2 = 000000	AMTYP4 = 000000	BPTOA 030660	DH134 135671	EM101 132626
ACPUOP = 000000	APASS = 000000	BPTOB 030666	DH24 135105	EM102 132661
ACTCHS 002724	APRIOR = 000000	BPTVEC = 000014	DH27 135151	EM103 132743
ACO = 0000000	APTCSU = 000040	BTER 051362	DH4 134646	EM104 133016
AC1 = 0000001	APTENV = 000001	BTEXP 003100	DH41 135214	EM105 133066
AC2 = 0000002	APTSIZ = 000200	BTGO 051162	DH43 135265	EM106 133141
AC3 = 0000003	APTSPO = 000100	BTRES 003110	DH47 135365	EM107 133217
AC4 = 0000004	ASWREG = 000000	BTTST 051662	DH5 134733	EM11 127275
AC5 = 0000005	ATESTN = 000000	BTTSTE 051710	DH65 135452	EM110 133253
AC6 = 0000006	AUNIT = 000000	BYPARR 106114	DH7 135032	EM111 133315
AC7 = 0000007	AUSWR = 000000	CCHPAS 003032	DH72 135516	EM112 133373
ADDLSB 114154	AVECT1 = 000000	CCR = 177746	DISPLA 001142	EM113 133456
ADDT 052264	AVECT2 = 000000	CCRTBL 111106	DISPRE 000174	EM114 133542
ADDTRP 137720	A126 025246	CHECK 015342	DHAPAR 124040	EM115 133563
ADDW0 = 000000	BA 002670	CHECK1 015420	DMARD 137452	EM116 133633
ADDW1 = 000000	BACDAT 105606	CHECK2 015476	DMATRN 137370	EM117 133714
ADDW10 = 000000	BA2 002710	CHECK7 053606	DPAREN 124100	EM12 127317
ADDW11 = 000000	BCR = 177524	CHEC10 054032	DSWR = 177570	EM120 133757
ADDW12 = 000000	BCSR = 177520	CHEC26 057214	DT1 136026	EM121 134025
ADDW13 = 000000	BDR = 177524	CHEC27 057556	DT105 136304	EM122 134123
ADDW14 = 000000	BFA 052316	CHEC30 060122	DT115 136314	EM123 134151
ADDW15 = 000000	BFAC1 052060	CHEC32 060554	DT130 136330	EM124 134203
ADDW2 = 000000	BFAC2 052104	CHEC33 061050	DT134 136336	EM125 134275
ADDW3 = 000000	BFAC3 052130	CHK7 053612	DT14 136074	EM126 134334
ADDW4 = 000000	BFAC4 052154	CHK10 054050	DT17 136106	EM127 134371
ADDW5 = 000000	BFAC5 052202	CHK7 053620	DT24 136120	EM13 127345
ADDW6 = 000000	BFAE 052356	CH10 054036	DT27 136130	EM130 134433
ADDW7 = 000000	BFB 052354	CKSWR = 104407	DT35 136142	EM131 134460
ADDW8 = 000000	BGNTLP 113472	CMPRTN 066176	DT4 136034	EM132 134510
ADDW9 = 000000	BIT0 = 000001	COUNT 003120	DT41 136154	EM133 134546
ADEVCT = 000000	BIT00 = 000001	CPEREG = 177766	DT43 136164	EM134 134567
ADEVM = 000000	BIT01 = 000002	CR = 000015	DT47 136200	EM14 127374
AENV = 000000	BIT02 = 000004	CRLF = 000200	DT5 136046	EM15 127417
AENVM = 000000	BIT03 = 000010	CSR1 002664	DT50 136212	EM16 127453
AFATAL = 000000	BIT04 = 000020	CSR12 002704	DT51 136224	EM17 127520
ALLCTR 003152	BIT05 = 000040	CSR2 002666	DT52 136236	EM2 127036
ALLEND 125304	BIT06 = 000100	CSR22 002706	DT64 136250	EM20 127564
ALROTS 021532	BIT07 = 000200	CURADD 114170	DT65 136262	EM21 127617
ALR1TS 021610	BIT08 = 000400	CURDAT 114162	DT7 136062	EM22 127662
ALR2TS 021666	BIT09 = 001000	C121A 023262	DT75 136272	EM23 127721
ALR3TS 021744	BIT1 = 000002	C121B 023302	DVDSUB 070236	EM24 127775
ALR4TS 022022	BIT10 = 002000	C121C 023322	DVFSUB 067244	EM25 130040
ALR5TS 022100	BIT11 = 004000	DAPABO 107350	D1 052726	EM26 130100
AMADR1 = 000000	BIT12 = 010000	DATA 002674	D2 052740	EM27 130147
AMADR2 = 000000	BIT13 = 020000	DATA2 002714	D3 052742	EM3 127050

Symbol table

EM30	130207	EXBAD2	115266	GPR4TS	005374	KIPAR4=	172350	MBTCC	015614
EM31	130261	EXITST	114134	GPR5TS	005452	KIPAR5=	172352	MBTD	032760
EM32	130306	EXPBDT	111126	GPR6TS	005530	KIPAR6=	172354	MBTE	051712
EM33	130347	EXPDAT	114146	GTSWR =	104406	KIPAR7=	172356	MBTF	032770
EM34	130411	EXPIR1	034344	HITMIS=	177752	KIPDR0=	172300	MBTO	033002
EM35	130451	EXPTBL	105634	HMPARR	105404	KIPDR1=	172302	MBTOA	033032
EM36	130477	EXPWDT	111116	HOP10	067506	KIPDR2=	172304	MBTOB	033034
EM37	130543	EXTFPA	140172	HOP11	070500	KIPDR3=	172306	MBTOC	033046
EM4	127062	FINNOP	021252	HOP12	071536	KIPDR4=	172310	MBTOD	033072
EM40	130605	FIN1	052444	HOP13	072564	KIPDR5=	172312	MBTOE	033074
EM41	130652	FIN10	054114	HOP14	073660	KIPDR6=	172314	MBTOF	033106
EM42	130736	FIN11	054220	HOP15	075354	KIPDR7=	172316	MBT1	051140
EM43	130764	FIN116	022742	HOP16	075670	KMCR =	177734	MBT2	051142
EM44	131015	FIN117	023056	HOP17	076352	LASTCH	116110	MBT2A	051154
EM45	131055	FIN120	023200	HOP18	100612	LCOSUB	101102	MBT8	051370
EM46	131137	FIN121	023574	HOP19	101230	LCFSUB	100464	MBT8A	051424
EM47	131227	FIN122	024044	HOP20	102650	LDPARS	136736	MBT8B	051452
EM5	127122	FIN125	025206	HOP21	103734	LDPDRS	136766	MBT8C	051500
EM50	131314	FIN126	026246	HOP22	104376	LEDS	123016	MBT8D	051526
EM51	131337	FIN127	026706	HOP44	066324	LF =	000012	MBT8E	051556
EM52	131371	FIN13	054434	HT =	000011	LKS =	177546	MBT8F	051606
EM53	131427	FIN130	027576	ILAOA	030760	LKSFL	002722	MBT8FG	051622
EM54	131463	FIN14	054604	ILBOB	030766	LKSINT	137062	MBT8I	051646
EM55	131521	FIN15	055022	ILL	053230	LOOP	004740	MB66	011770
EM56	131555	FIN16	055212	ILLBOA	031054	LOOPIN	003154	MCB44	011634
EM57	131601	FIN17	055424	ILLBOB	031062	LOST	052424	MCLRD	076416
EM6	127155	FIN2	052544	ILLOP1	053030	LOWADD	003016	MCLRI	076504
EM60	131633	FIN20	055622	ILLOP2	053124	LSTADD	114166	MCMPD	066014
EM61	131701	FIN21	056042	INITMM	136574	LXPSUB	102400	MCTSCC	016066
EM62	131730	FIN22	056226	INQ22	125446	MACCC	016216	MDAO	010406
EM63	132001	FIN23	056362	INTERR	110106	MACE	051322	MDCCC	016004
EM64	132032	FIN24	056562	INTRPC	110052	MAC0	051212	MDDSUB	075054
EM65	132071	FIN26	057316	IOTOA	030366	MACOA	051216	MDFSUB	073350
EM66	132141	FIN27	057660	IOTOB	030374	MAC1	051226	MDIVD	067506
EM67	132200	FIN30	060224	IOTVEC=	000020	MAC2	051242	MDIVF	066324
EM7	127220	FIN31	060364	IOXX	033520	MAC3	051256	MDMO	010362
EM70	132235	FIN32	060656	J11FLT	140257	MAC4	051272	MDM27	010472
EM71	132266	FIN33	061152	KOPAR0=	172360	MAC5	051306	MDSO	010436
EM72	132322	FIN4	053134	KOPAR1=	172362	MADCC	016142	MEMK	123434
EM73	132360	FIN5	053240	KOPAR2=	172364	MAIREG=	177750	MEMQ	123464
EM74	132404	FIN6	053440	KOPAR3=	172366	MALCC	016770	MEMTO	030420
EM75	132435	FIN7	053664	KOPAR4=	172370	MARCC	017050	MET	032442
EM76	132466	FLAG	003030	KOPAR5=	172372	MASK	003160	META	032500
EM77	132536	FLO	003062	KOPAR6=	172374	MA11	011010	METB	032510
ENUHRT	105576	FLOAT	003052	KOPAR7=	172376	MA55	011714	METD	032520
ENDLUP	114110	FMPARR	105214	KOPDR0=	172320	MBB11	011106	METF	032530
ENDMOV	114172	FPAFLT	140174	KOPDR1=	172322	MBB22	011370	METO	032550
ENDTAG	115314	FPAOPT	140160	KOPDR2=	172324	MBCCC	015664	METOA	032614
ENDTLP	114060	FPVEC =	000244	KOPDR3=	172326	MBC00	010612	METOB	032624
ERR	140060	FRSTST	004742	KOPDR4=	172330	MBC11	011166	METOC	032636
ERRCNT	003022	FSTADD	114164	KOPDR5=	172332	MBC22	011456	METOD	032662
ERRFP	140056	FWDSEQ	114152	KOPDR6=	172334	MBI00	010532	METOE	032672
ERROR =	104000	GOODAD	003020	KOPDR7=	172336	MBPT0	030614	METOF	032704
ERROUT	104536	GPROTS	005104	KIPAR0=	172340	MBSCC	015736	MFA	051714
ERRVEC =	000004	GPR1TS	005162	KIPAR1=	172342	MBT	032724	MFACU	051712
ERTYPE	136356	GPR2TS	005240	KIPAR2=	172344	MBTA	032746	MFSRCM	061154
EXBAD	114124	GPR3TS	005316	KIPAR3=	172346	MBTB	032750	MIALL	033320

Symbol table

MIALLA	033344	MJR77A	014776	MLDC2	077644	MRTB	031550	MST6	010240
MIALLB	033346	MJSI	033400	MLDDM2	061240	M RTE	031560	MST7	010306
MIALLD	033356	MJSIA	033432	MLDDM3	061342	MRTF	031570	MSW37	012614
MIALLF	033366	MJSIB	033434	MLDDM4	061476	MRTD	031602	MSXP	104076
MIL	033120	MJSIC	033446	MLDDM5	061616	MRTOA	031632	MSXT	017302
MILA	033144	MJSID	033472	MLDDM6	061722	MRTOB	031634	MSXTCC	017140
MILAO	030712	MJSIE	033474	MLDDM7	062016	MRTOC	031646	MS11	011046
MILB	033146	MJSIF	033506	MLDM27	062120	MRTOD	031672	MS22	011320
MILD	033156	MJSR	013716	MLDSUB	072322	MRTOE	031674	MS33	011550
MILF	033166	MJSRA	014034	MLFSUB	071274	MRTOF	031706	MS77	012040
MILLBO	031012	MJSRB	014200	MLS1	076550	MRTS	015002	MTP	031720
MILLO	030224	MJSRC	014346	MLS2	076662	MSB	064216	MTPA	032122
MILLOA	030270	MJSR1	014036	MLS3	076774	MSBCC	016416	MTPAA	032176
MILLOB	030276	MJSR1A	014050	MLS4	077124	MSBCCC	016466	MTPAE	032226
MILO	033200	MJSR1B	014074	MLS5	077240	MSCD	102650	MTPAH	032174
MILOA	033232	MJSR2	013754	MLS6	077370	MSCF	103734	MTPAL	032164
MILOB	033234	MJSR2A	013766	MLS7	077510	MSDF	075670	MTPB	031752
MILOC	033246	MJSR2B	014012	MLXP	101230	MSER	177744	MTPF	031772
MILOD	033272	MJSR3	014202	M MARK	017554	MSFD	075354	MTPQ	032004
MILOE	033274	MJSR3A	014214	M MODD	073660	MSFDI	076352	MTPQA	032034
MILOF	033306	MJSR3B	014240	M MODF	072564	MSOB	017500	M:POB	032036
MIOT	032246	MJSR4	014116	M MRL5	012452	MSPAA	007640	MTPOC	032050
MIOTA	032270	MJSR4A	014130	M MRO	177572	MSPAU	027576	MTPOD	032074
MIOTB	032272	MJSR4B	014154	M MR1	177574	MSPB	005732	MTPOE	032076
MIOTD	032302	MJSR5	014350	M MR2	177576	MSPBB	007724	MTPOF	032110
MIOTF	032312	MJSR5A	014362	M MR3	172516	MSPC	005756	MTPQ	031762
MIOTO	030322	MJSR5B	014406	M MU	137552	MSPD	006006	MTPR	031750
MITO	032324	MJSR6	014264	M MULD	071536	MSPEO	006044	MTRPO	030516
MITOA	032354	MJSR6A	014276	M MULF	070500	MSPF	006106	MTRY	027740
MITOB	032356	MJSR6B	014322	M MUTRP	137724	MSPG	006150	MTRYA	030014
MITOC	032370	MJSR7	014432	M MVCC	015550	MSPH	006206	MTRYB	030034
MITOD	032414	MJSR7A	014444	M MVEC	000250	MSPI	006252	MTRYM	030066
MITOE	032416	MJSR7E	014470	M M11	010700	MSPJ	006312	MTSO	015550
MITOF	032430	MJU1	012772	M M22	011246	M SPL	006432	MTT	031176
MJ	012720	MJU1A	013004	M NCCCC	016300	MSPM	006500	MTTA	031232
MJP	013556	MJU2	012734	M NGOP	063406	MSPN	006556	MTTB	031234
MJP17	013566	MJU2A	012746	M NARM1	062226	MSPQ	006624	MTTD	031244
MJP27	013610	MJU2B	012756	M NARM2	062506	MSPR	007026	MTTE	031254
MJP27A	013622	MJU3	013016	M NARM3	062660	MSPS	007134	M TTR	031360
MJP37	013674	MJU3A	013030	M NARM4	063022	MSPV	007222	MTTRA	031424
MJP37A	013706	MJU3B	013040	M NRM	064360	MSPX	007270	MTTRB	031426
MJP67	013632	MJU4	013112	MODE1	046034	MSPY	007306	MTTRC	031440
MJP67A	013644	MJU4A	013124	MODE2	047414	MSPZ	007406	MTTRD	031476
MJP67B	013660	MJU4B	013136	MODGAR	073650	MSPV0	007352	MTTRE	031500
MJP77	013662	MJU5	013052	MRLB1	012200	MSPX	007452	MTTRF	031512
MJP77E	013716	MJU5A	013064	MRLCC	016552	MSPY	007522	MTTS	031266
MJRA	014474	MJU5B	013076	MRL0	012126	MSPZ	007570	MTTSA	031322
MJR27	014530	MJU6	013152	MRL2	012260	MSP0	005710	MTTSB	031326
MJR27A	014564	MJU6A	013164	MRL3	012336	MSTB3	007774	MTTSD	031336
MJR27B	014604	MJU7	013204	MRL4	012412	MSTO	031102	MTTSE	031346
MJR37	014666	MJU7E	013216	MRL6	012514	MSTOE	031156	MTTSQ	031324
MJR37A	014722	MJ2	012770	MRL7	012552	MUVAD	064500	MXDF1	063142
MJR6A	014662	MJ5	013150	MRRB1	012672	MST4	010046	MXOR	017420
MJR6B	014664	MJ7	013200	MRRCC	016632	MST4B	010104	MXRCC	017210
MJR67	014606	MLCD	100612	MRR0	012650	MST5	010144	M2	004766
MJR67A	014642	MLCF	077760	MRT	031524	MST5B	010176	M3	005002
MJR77	014742	MLCC	065376	MRTA	031546				

Symbol table

M4	005022	POLY	= 120001	SDPAR5	= 172272	SW01	= 000002	TAB42	003724
M5	005042	PROCNT	023126	SDPAR6	= 172274	SW02	= 000004	TAB43	003734
M6	005062	PRO	= 000000	SDPAR7	= 172276	SW03	= 000010	TAB45	003744
NEWADD	003026	PR1	= 000040	SDPDR0	= 172220	SW04	= 000020	TAB46	003754
NEWDAT	114160	PR2	= 000100	SDPDR1	= 172222	SW05	= 000040	TAB47	003764
NOABRT	141356	PR3	= 000140	SDPDR2	= 172224	SW06	= 000100	TAB47A	003774
NOFPA	140166	PR4	= 000200	SDPDR3	= 172226	SW07	= 000200	TAB48	004004
NOTOK	140002	PR5	= 000240	SDPDR4	= 172230	SW08	= 000400	TAB49	004014
NULL	= 000000	PR6	= 000300	SDPDR5	= 172232	SW09	= 001000	TAB5	003274
NXMF IN	047636	PR7	= 000340	SDPDR6	= 172234	SW1	= 000002	TAB5A	003304
NXMPAR	111350	PS	= 177776	SDPDR7	= 172236	SW10	= 002000	TAB6	003314
NXMTRP	047244	PSW	= 177776	SEQ	003072	SW11	= 004000	TAB6A	003324
ODDX	033620	PSWBTS	005612	SFDSUB	075550	SW12	= 010000	TAB7	003334
OK	137760	PWRVEC	= 000024	SIMG0A	002702	SW13	= 020000	TAB8	003344
OKAY7	043056	Q22EN	003002	SIPAR0	= 172240	SW14	= 040000	TAB9	003354
OKAY7A	043066	Q22INT	137350	SIPAR1	= 172242	SW15	= 100000	TAPAB0	107514
OKA7	043044	Q22SIZ	137070	SIPAR2	= 172244	SW2	= 000004	TA114	021170
OK1	140006	RAMPAR	137012	STPAR3	= 172246	SW3	= 000010	TA116	022644
OK7	043026	RBUF	= 177562	SIPAR4	= 172250	SW4	= 000020	TBITVE	= 000014
ONOQ22	137314	RCSR	= 177560	SIPAR5	= 172252	SW5	= 000040	TB114	021200
PAR	137674	RDCHR	= 104410	SIPAR6	= 172254	SW6	= 000100	TC114	021210
PARAD1	045664	RDLIN	= 104411	SIPAR7	= 172256	SW7	= 000200	TD114	021220
PARAD2	047276	RDOCT	= 104412	SIPDR0	= 172200	SW8	= 000400	TEMP	002740
PARVA1	045716	RECDAT	114150	SIPDR1	= 172202	SW9	= 001000	TE102	017760
PARVA2	047330	RECDST	003142	SIPDR2	= 172204	SXPSUB	104256	TE103	020002
PARVA3	047446	RECFEC	003122	SIPDR3	= 172206	TAB1	003234	TE104	020024
PAR1	137676	RECST	003132	SIPDR4	= 172210	TAB10	003364	TE105	020046
PCR	= 177522	RESVEC	= 000010	SIPDR5	= 172212	TAB11	003374	TE106	020070
PDR	137652	RET1	015266	SIPDR6	= 172214	TAB11A	003404	TE107	020112
PDR1	137654	RET2	015354	SIPDR7	= 172216	TAB12	003414	TE110	020134
PHY1	046002	RET3	015432	SIXBIT	115460	TAB13	003424	TE111	020156
PIR	= 177772	RITEDA	114156	SLEND	122620	TAB13B	003434	TE112	020200
PIRQ	= 177772	RTSE	015054	SLOC00	003012	TAB14	003444	TE113	020422
PIRQNX	034222	RTS1	015022	SLOC01	003014	TAB15	003454	TE113A	020642
PIRQT	125736	RTS6	015032	SPAU1	027614	TAB16	003464	TE114	021014
PIRQVE	= 000240	RXXY	034040	SPAU2	027636	TAB17	003474	TE115	022164
PIRRTN	034522	R6	=#000006	SPAU3	027656	TAB18	003504	TE115A	022374
PIRTBL	034324	R7	=#000007	SPAU4	027702	TAB2	003244	TE115B	022404
PIRTEX	034224	SAVBR	002730	SPAU5	027716	TAB21	003514	TE115C	022412
PIRXXX	034166	SAVMRO	003042	SPAU6	027734	TAB22	003524	TE115D	022420
PIR1	034224	SAVMR1	003044	SPS	003074	TAB23	003534	TE115F	022426
PIR2	034344	SAVMR2	003046	SPSJ	003076	TAB24	003544	TE116	022430
PIR2EX	034452	SAVPCR	002726	SRO	= 177572	TAB25	003554	TE116A	022704
PIR3	034452	SAVPOS	003156	SR1	= 177574	TAB26	003564	TE116B	022710
PIR3EX	034546	SAVSUP	003036	SR2	= 177576	TAB27	003574	TE116C	022722
PIR4	034546	SAVSWR	003050	SR3	= 172516	TAB28	003604	TE116D	022732
PIR5	034624	SAVUSE	003040	STACK	= 001100	TAB29	003614	TE117	022742
PIR5EX	035034	SAV30	004112	START	004024	TAB29A	003624	TE117A	023054
PIR6	035034	SAV32	004114	STBOT	= 001000	TAB3	003254	TE120	023060
PIR6EX	035136	SCDSUB	103470	STKLMT	= 177774	TAB30	003634	TE120A	023134
PITBL1	034430	SCOPE	= 000004	STMOVI	113326	TAB31	003644	TE121	023210
PITBL2	035020	SDFSUB	076120	STMOVT	114450	TAB32	003654	TE122	023574
PI1	034474	SDPAR0	= 172260	SUBT	052232	TAB33	003664	TE123	024046
PI2	034506	SDPAR1	= 172262	SWR	001140	TAB34	003674	TE124	024266
PI3	034510	SDPAR2	= 172264	SWREG	000176	TAB4	003264	TE125	024520
PLFO	043674	SDPAR3	= 172266	SWO	= 000001	TAB40	003704	TE125A	024760
PLF1	044066	SDPAR4	= 172270	SW00	= 000001	TAB41	003714	TE126	025206

Symbol table

TE126A	025254	TSFP5	053134	TST13	107044	TS14	047134	T123D	024254
TE126B	025730	TSFP6	053240	TST14	107360	TS15	050020	T123E	024256
TE127	026246	TSFP7	053440	TST15	107524	TS16	051024	T123F	024262
TE127A	026446	TSF10	054054	TST16	107752	TS16A	051016	T124A	024462
TE130	026710	TSF13	054424	TST17	110156	TS1822	046066	T124B	024472
TE130A	027166	TSF14	054574	TST2	104376	TS26D0	057236	T124C	024500
TF114	021230	TSF15	055012	TST20	110402	TS26D1	057246	T124D	024506
TG114	021240	TSF16	055202	TST21	111150	TS26D2	057256	T124E	024510
THRBIT	115726	TSF17	055414	TST22	111360	TS26D3	057266	T124F	024514
TH114	021250	TSF2	052514	TST23	111600	TS26D4	057276	T13FIN	046066
TIMDEL	120602	TSF20	055612	TST24	112136	TS26D5	057306	T14	047160
TIMEOU	053114	TSF21	056032	TST25	112714	TS27D0	057600	T14FIN	047532
TIMOUT	003000	TSF22	056216	TST26	113056	TS27D1	057610	T15	050070
TKVEC	000060	TSF23	056352	TST27	114172	TS27D2	057620	T15A	050146
TMM16A	050534	TSF24	056552	TST3	104542	TS27D3	057630	T15FIN	050174
TMM16B	050414	TSF31	060354	TST30	115314	TS27D4	057640	UDPAR0-	177660
TMM16C	050444	TSF6	053430	TST31	115466	TS27D5	057650	UDPAR1-	177662
TMM16D	050474	TSF7	053624	TST32	115734	TS30D0	060144	UDPAR2-	177664
TMM16E	050524	TSL00P	114556	TST33	116254	TS30D1	060154	UDPAR3-	177666
TMM16F	050526	TSM	043116	TST34	116406	TS30D2	060164	UDPAR4-	177670
TM16A	051120	TSMB	043136	TST35	116772	TS30D3	060174	UDPAR5-	177672
TOUT	137542	TSMC	043146	TST36	117156	TS30D4	060204	UDPAR6-	177674
TPVEC	000064	TSMU0	015056	TST37	117464	TS30D5	060214	UDPAR7-	177676
TRAPVE	000034	TSMU1	035152	TST4	104642	TS32D0	060576	UDPDR0-	177620
TRPFLG	140054	TSMU2	035236	TST40	117632	TS32D1	060606	UDPDR1-	177622
TRPOA	030562	TSMU3	036436	TST41	120134	TS32D3	060626	UDPDR2-	177624
TRPOB	030570	TSMU4	036736	TST42	120302	TS32D4	060636	UDPDR3-	177626
TRTVEC	000014	TSMU5	037666	TST43	120612	TS32D5	060646	UDPDR4-	177630
TRYMA	030126	TSMU6	040020	TST44	120652	TS33D0	061072	UDPDR5-	177632
TRYMB	030160	TSMU7	042524	TST45	120750	TS33D1	061102	UDPDR6-	177634
TRYMC	030206	TSMU8	043152	TST46	121032	TS33D2	061112	UDPDR7-	177636
TSD31	060252	TSMU9	043354	TST47	121246	TS33D3	061122	UFDLFG	004116
TSEND	115276	TSMU10	044254	TST5	105226	TS33D4	061132	UFDSET	000001
TSFP1	052356	TSMU11	044636	TST50	121300	TS33D5	061142	UIPAR0-	177640
TSFP10	053664	TSMU12	045074	TST51	121644	TS6DA	053410	UIPAR1-	177642
TSFP11	054114	TSMU13	045462	TST52	122112	TS6DAT	053420	UIPAR2-	177644
TSFP12	054220	TSMU14	046530	TST53	122402	TS7	042774	UIPAR3-	177646
TSFP13	054270	TSMU15	047654	TST54	122620	TS7DA1	053634	UIPAR4-	177650
TSFP14	054434	TSMU16	050174	TST55	123142	TS7DA2	053644	UIPAR5-	177652
TSFP15	054606	TSMU6A	040126	TST56	123516	TS7DA4	053654	UIPAR6-	177654
TSFP16	055024	TSMU6B	040564	TST57	123764	TS7FIN	043150	UIPAR7-	177656
TSFP17	055214	TSMU6C	041324	TST6	105412	TS9FIN	044252	UIPDR0-	177600
TSFP2	052446	TSMU6D	042024	TST60	124136	TYPDS	104405	UIPDR1-	177602
TSFP20	055426	TSM16A	050270	TST61	124336	TYPE	104401	UIPDR2-	177604
TSFP21	055624	TSM16B	050316	TST62	125304	TYPOC	104402	UIPDR3-	177606
TSFP22	056044	TSM16C	050342	TST63	125562	TYPON	104404	UIPDR4-	177610
TSFP23	056230	TSM16D	050406	TST64	125746	TYPOS	104403	UIPDR5-	177612
TSFP24	056364	TSM7	043104	TST65	126046	T10FIN	044636	UIPDR6-	177614
TSFP25	056564	TSM9	043510	TST66	126230	T11FIN	045074	UIPDR7-	177616
TSFP26	056722	TSTADD	003024	TST7	105652	T114	021146	UNXPIR	034450
TSFP27	057320	TSTEND	116770	TS10	044554	T116	022612	UQUIET	004120
TSFP3	052544	TSTLOC	003162	TS10D1	054064	T12FIN	045460	VIREOP	126756
TSFP30	057662	TSTLUP	113410	TS10D2	054074	T122A	023672	VIR1	045750
TSFP31	060226	TST1	004740	TS10D4	054104	T122B	024024	VIR2	047362
TSFP32	060366	TST10	106126	TS11	044754	T123A	024230	VIR3	047500
TSFP33	060660	TST11	106370	TS11D1	054210	T123B	024240	VMKOR	004122
TSFP4	052660	TST12	106644	TS12	045412	T123C	024246	VQBE1	002676

Symbol table

VQBE2	002716	\$DDW10	001310	\$ETEND	001324	\$MSGTY	001200	\$TIMES	001164
VQPR1	002700	\$DDW11	001312	\$FATAL	001202	\$MSWR	143376	\$TKB	001146
VQPR2	002720	\$DDW12	001314	\$FFLG	141624	\$MTYP1	001231	\$TKS	001144
WC	002672	\$DDW13	001316	\$FILLC	001156	\$MTYP2	001235	\$TMP0	001160
WC2	002712	\$DDW14	001320	\$FILLS	001155	\$MTYP3	001241	\$TMP1	001162
WLDTRP	140044	\$DDW15	001322	\$GDADR	001120	\$MTYP4	001245	\$TN	= 000067
XBUF	= 177566	\$DDW2	001270	\$GDDAT	001124	\$MXCNT	140774	\$TPB	001152
XCBIT	016710	\$DDW3	001272	\$GET42	140430	\$NULL	001154	\$TPFLG	001157
XCSR	= 177564	\$DDW4	001274	\$GTSMR	142704	\$NWTST	= 000000	\$TPS	00115L
XME100	017714	\$DDW5	001276	\$HD	= 000001	\$OCNT	142404	\$TRAP	143522
XME101	017736	\$DDW6	001300	\$HIBTS	000232	\$OMODE	142406	\$TRAP2	143544
\$APTHD	000232	\$DDW7	001302	\$HIOCT	143520	\$OVER	140752	\$TRP	= 000013
\$ATYC	141404	\$DDW8	001304	\$ICNT	001104	\$PASS	001206	\$TRPAD	143556
\$ATY1	141360	\$DDW9	001306	\$ILLUP	143744	\$PASTM	000240	\$TSTM	000236
\$ATY3	141366	\$DEVCT	001210	\$INTAG	001135	\$POWER	143752	\$TSTNM	001102
\$ATY4	141376	\$DEVM	001256	\$ITEMB	001114	\$PWRDN	143604	\$TTYIN	143354
\$AUTOB	001134	\$DOAGN	140450	\$LF	001176	\$PWRMG	143740	\$TYPDS	142410
\$BASE	001254	\$DTBL	142614	\$LFLG	141623	\$PWRUP	143656	\$TYPE	141626
\$BDADR	001122	\$ENDAD	140440	\$LPADR	001106	\$QUES	001174	\$TYPEC	142040
\$BDDAT	001126	\$ENDCT	140406	\$LPERR	001110	\$RDCHR	143116	\$TYPEX	142160
\$BELL	001170	\$ENDMG	140457	\$MADR1	001232	\$RDLIN	143246	\$TYPC	142206
\$CDW1	001260	\$ENULL	140454	\$MADR2	001236	\$RDOCT	143420	\$TYPON	142222
\$CDW2	001262	\$ENV	001220	\$MADR3	001242	\$RDSZ	= 000010	\$TYPOS	142162
\$CHARC	142156	\$ENVH	001221	\$MADR4	001246	\$RTNAD	140452	\$UNIT	001212
\$CKSMR	142634	\$EOP	140340	\$MAIL	001200	\$SAVR6	143750	\$UNITM	000242
\$CMTAG	001100	\$EOPCT	140400	\$MAMS1	001230	\$SCOPE	140474	\$USWR	001224
\$CM3	= 000000	\$ERFLG	001103	\$MAMS2	001234	\$SETUP	= 000137	\$VECT1	001250
\$CM4	= 000002	\$ERMAX	001115	\$MAMS3	001240	\$STUP	= 177777	\$VECT2	001252
\$CNTLG	143371	\$ERROR	140776	\$MAMS4	001244	\$SVLAD	140716	\$XOFF	= 000023
\$CNTLU	143364	\$ERRPC	001116	\$MBADR	000234	\$SVPC	= 000232	\$XON	= 000021
\$CPUOP	001226	\$ERRTB	001324	\$MFLG	141622	\$SWR	= 167400	\$XTSTR	140514
\$CRLF	001175	\$ERTTL	001112	\$MNEW	143407	\$SWREG	001222	\$GET4	= 000000
\$DBLK	142624	\$ESCAP	001166	\$MSGAD	001214	\$SWRMK	= 000300	\$OFILL	142405
\$DDW0	001264	\$ETABL	001220	\$MSGLG	001216	\$TESTN	001204	.\$X	= 000232
\$DDW1	001266								

. ABS. 143762 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 10

*** Assembler statistics

Work file reads: 471
 Work file writes: 403
 Size of work file: 63968 Words (250 Pages)
 Size of core pool: 19714 Words (75 Pages)
 Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:24:42.02
 COKDAEO.COKDAEO/NL:TOC/-SP=ORION.MLB/ML,COKDAEO.MAC/DS:GBL