

KD11-Z

11/24/44 UBI MAP
CKKUAE0

AH-F629E-MC
FICHE 1 OF 1

JUL 1982
COPYRIGHT © 79-82
MADE IN USA



A large grid of microfiche frames, each containing a small document page. The pages are arranged in approximately 15 rows and 12 columns. Each frame contains a document page with text and some graphical elements, though the text is too small to read clearly. The entire grid is set against a dark background.



.REM @

IDENTIFICATION

1
2
3
4
5
6
7
8
9
10
11 PRODUCT CODE: AC-F627E-MC
12 PRODUCT NAME: CKKUAE0 11/24/44 UBI MAP
13
14 DATE CREATED: JANUARY, 1982
15
16 MAINTAINER: DIAGNOSTIC ENGINEERING
17
18 AUTHOR: DAN MILLEVILLE
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

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

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

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

COPYRIGHT (C) 1979, 1982 BY DIGITAL EQUIPMENT CORPORATION

39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

HISTORY SECTION

CKKUAA0 WAS RELEASED OCTOBER, 1979 > INITIAL RELEASE
CKKUAB0 WAS RELEASED OCTOBER, 1980 > ADDITION OF UNIBUS MEMORY TEST
CKKUAC0 WAS RELEASED APRIL, 1981 > TESTING OF 11/24 MAP, 11/24 WITH UNIBUS
MEMORY ONLY, & POWER MONITOR BIT CHECK.
CKKUAD0 WAS RELEASED OCTOBER, 1981 > SETUP OF SOFTWARE/HARDWARE SWR ADDRESS
WAS ADDED TO STARTUP CODE.
> WHEN SELECTED, THE OPTIONAL CACHE TESTS
NOW RUN CORRECTLY.
CKKUAE0 WAS RELEASED JANUARY, 1982 > THE LMA TEST (11/24 ONLY) WAS MODIFIED TO
DO A MORE THOROUGH JOB OF CHECKING THE
ADDRESS LINES.

TABLE OF CONTENTS

56	
57	
58	1) ABSTRACT
59	
60	2) REQUIREMENTS
61	
62	2.1 EQUIPMENT
63	2.2 STORAGE
64	2.3 PRELIMINARY PROGRAMS
65	
66	3) LOADING PROCEDURE
67	
68	3.1 METHOD
69	
70	4) STARTING PROCEDURE
71	
72	4.1 STARTING ADDRESS
73	4.2 PROGRAM AND OPERATOR ACTION
74	4.3 SPECIAL STARTING PROCEDURE
75	
76	5) OPERATING PROCEDURE
77	
78	5.1 OPERATIONAL SWITCH SETTINGS
79	5.2 SUB-ROUTINE ABSTRACTS
80	5.3 RUNNING UNDER APT
81	
82	6) ERRORS
83	
84	6.1 ERROR HALTS AND DESCRIPTION
85	6.2 ERROR RECOVERY
86	6.3 SAMPLE ERROR MESSAGES
87	
88	7) RESTRICTIONS
89	
90	7.1 STARTING RESTRICTIONS
91	7.2 OPERATING RESTRICTIONS
92	
93	8) MISCELLANEOUS
94	
95	8.1 EXECUTION TIME
96	8.2 ADDRESS GENERATION IN THE PDP-11/44
97	
98	9) PROGRAM DESCRIPTION

99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP 11/24 OR 11/44 ON WHICH THE CPU, CACHE (IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTIC PROGRAMS HAVE BEEN RUN. THE PROGRAM WILL DETECT ALL ERRORS THAT ORIGINATE WITH THE MAP BOX AND PROVIDE LOOPING CAPABILITIES SO THAT THE FIELD SERVICE ENGINEER CAN VERIFY THE FAILURES. THERE MAY BE SOME CASES, SUCH AS THE CACHE REGISTER DATA PATH, AND CACHE MEMORY DATA PATH, WHERE INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE ENGINEER CAN COMPLETE THE ISOLATION PROCESS.

IF THE PROGRAM CATCHES AN ERROR IN AN EARLY TEST AND IS ALLOWED TO CONTINUE RUNNING THROUGH THE LATER TESTS THE ERROR INDICATIONS FROM THOSE LATER TESTS MAY BE INVALID. THIS IS DUE TO THE STRUCTURE OF THE PROGRAM, WHICH ASSUMES THAT ALL AREAS TESTED PRIOR TO THE CURRENT TEST ARE FUNCTIONING PROPERLY.

THE ERROR TYPE OUTS WILL BE IN TABLE FORMAT, WITH A MESSAGE INDICATING

THE CLASS OF ERROR, A HEADER IDENTIFYING EACH COLUMN AND A REPORT OF ALL PERTINENT DATA. WHEN THE TEST CAN PRODUCE MORE THAN ONE ERROR CONDITION, A SUMMARY OF ERRORS WILL BE GIVEN AT THE END OF THAT TEST CONSISTING OF: THE LOGICAL 'AND' AND 'OR' OF THE DATA PREVIOUSLY REPORTED AND THE NUMBER OF ERRORS IN THIS TEST. (SEE SECTION 6.3 FOR AN EXAMPLE OF THE ERROR TYPEOUTS.)

126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/24 OR 44 COMPUTER, INCLUDING THE CPU, CACHE (11/44 ONLY), MEMORY MANAGEMENT, AND AN LA-30 OR EQUIVALENT DEVICE FOR ERROR MESSAGES.

2.2 STORAGE

THIS PROGRAM WILL REQUIRE 8K TO LOAD BUT WILL UTILIZE ALL EXISTING CORE FOR A DUAL ADDRESSING TEST OF MEMORY FROM THE UNIBUS.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE (IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. THE MEMORY DIAGNOSTIC SHOULD AT LEAST MAKE A QUICK VERIFY OF THE AREA OF MEMORY THIS PROGRAM WILL LOAD AND RUN IN.

146
147
148
149
150
151
152

3. LOADING PROCEDURE

3.1 METHOD.

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS
SUPPORTED BY XXDP AND SHOULD BE LOADED USING THE XXDP
PROCEDURE FOR THAT DEVICE.

153 4. STARTING PROCEDURE
154
155 4.1 STARTING ADDRESS
156 PROGRAM STARTS AT ADDRESS 200
157
158 4.2 PROGRAM AND/OR OPERATOR ACTION
159
160 THE PROGRAM WILL IDENTIFY ITSELF. IF CPU IS AN 11/44, PROGRAM
161 WILL PROCEED WITH TESTING. IF CPU IS AN 11/24, AND LOCATION 176
162 CONTAINS ZERO (SUCH AS JUST AFTER LOADING), PROGRAM WILL PROMPT
163 FOR A SWITCH REGISTER INPUT. IF NONE IS REQUIRED, ENTER <CR>.
164 IF YOU DO SO, PROGRAM WILL DEPOSIT A '1' IN LOCATION 176 (MEANINGLESS
165 UNLESS BIT 8 IS ALSO SET) SO THAT SUBSEQUENT RUNNING OF THE PROGRAM
166 WILL NOT RESULT IN THE SAME REQUEST. IF MANUFACTURING IS RUNNING
167 THIS PROGRAM ON AN 11/24 WITH UNIBUS MEMORY ONLY (NO MAIN MEMORY),
168 ENTER A NUMBER WITH AT LEAST BIT 11 SET (SWR=4000). THIS WILL KEY
169 THE PROGRAM TO TEST THE 11/24 IN THAT CONFIGURATION.
170
171 4.3 SPECIAL STARTING PROCEDURE
172
173 IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND
174 YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN
175 WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL
176 REGISTER (17777746) WITH THE DESIRED NUMBER. THEN LOAD
177 THE PC (17777707) WITH THE STARTING ADDRESS (200) AND
178 PRESS "CONTINUE". THE PROGRAM WILL NOW RUN NORMALLY EXCEPT
179 THAT CERTAIN TESTS WILL BE SKIPPED SINCE THE CACHE IS DISABLED.
180 THIS FACT IS INDICATED IN THE ABSTRACT OF EACH TEST THAT
181 CHECKS THE CACHE CONTROL REGISTER.
182
183 DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:
184 BIT00 -DISABLE TRAPS
185 BIT02 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 0
186 BIT03 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 1
187 BIT09 -UNCONDITIONAL CACHE BYPASS
188

189 5. OPERATING PROCEDURE
 190
 191 5.1 OPERATIONAL SWITCH SETTINGS
 192 SW15 1= HALT ON ERROR
 193 SW14 1= LOOP ON TEST
 194 SW13 1= INHIBIT ERROR TYPEOUTS
 195 SW12 1= INHIBIT TRACE TRAP
 196 SW11 1= SET WHEN RUNNING AN 11/24 WITH NO MAIN MEMORY
 197 AND ONLY UNIBUS MEMORY
 198 SW10 1= BELL ON ERROR
 199 SW09 1= LOOP ON ERROR
 200 SW08 1= LOOP ON TEST IN SWR<05:00>
 201 SW07 1= INHIBIT MULTIPLE ERROR TYPE OUTS
 202 SW06 1= SELECT CACHE TESTS. THIS IS USED FOR
 203 MFG. QUICK VERIFY STATION AND CAN BE SELECTED
 204 BY APT SCRIPTING. THESE TESTS ASSUME THAT
 205 ALL MODULES EXCEPT UBI MODULE ARE KNOWN GOOD.
 206 SW05 1= SELECT UNIBUS MEMORY TESTS
 207 SET WHEN A WINDOW EXISTS (SOME UBMAP JUMPERS CUT), AND
 208 UNIBUS MEMORY OCCUPIES THE ENTIRE WINDOW ADDRESS AREA.
 209
 210 5.2 SUB-ROUTINE ABSTRACTS
 211 ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE BEFORE THEIR
 212 EXPANSION AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS.
 213 BELOW IS A LIST OF THE SUBROUTINE TITLES.
 214
 215 5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS) (SOME IN THIS SOURCE)
 216 SCOPE HANDLER ROUTINE
 217 ERROR HANDLER ROUTINE
 218 ERROR MESSAGE TYPE OUT ROUTINE
 219 CONVERT 16-BIT VIRTUAL ADDRESSES TO 22-BIT PHYSICAL ADDRESSES
 220 SAVE AND RESTORE R0-R5 ROUTINES
 221 TYPE ROUTINE
 222 BINARY TO OCTAL (ASCII) AND TYPE
 223 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 224 TRAP DECODER
 225 POWER DOWN AND UP ROUTINES
 226 DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
 227 END OF PASS ROUTINE
 228
 229 5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM
 230 SUBROUTINE TO TURN OFF AND SAVE T-BIT
 231 SUBROUTINE TO RESTORE T-BIT TO ITS PREVIOUS CONDITION
 232 SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
 233 SUBROUTINE TO EXTRACT MAP ADDRESS FROM PAR CONTENTS
 234 SUBROUTINE TO DETERMINE 11/24 WITH UNIBUS MEM ONLY & CHECK LMA'S IF SO
 235
 236 5.2.3 TRAP AND ABORT HANDLER ROUTINES
 237 CPU TRAP HANDLER ROUTINE
 238 CACHE TRAPS AND ABORTS HANDLER ROUTINE
 239 MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
 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
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297

5.3 RUNNING UNDER APT

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD. THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. E TABLE 'A' IS USED FOR APT DUMP MODE.
A. IN ADDITION TO NORMAL CPU DIAGNOSTIC TESTS, THIS TABLE WILL SELECT THE OPTIONAL CACHE TESTS, (\$SWREG=100).
2. E TABLE 'B' IS USED FOR APT QV MODE WHILE RUNNING ON A MANUFACTURING QV STATION. IT ACCOMPLISHES WHAT ETABLE 'A' DOES BUT ADDITIONALLY SUPPRESSES TYPEOUTS.(\$ENVM=240)
3. ETABLE 'C' IS USED FOR APT QV OR RUNTIME MODES WHILE RUNNING ON SYSTEMS OTHER THAN MFG. QV STATIONS. THIS TABLE DESELECTS THE OPTIONAL CACHE TESTS.
4. ETABLE 'D' IS USED BY MANUFACTURING TO RUN AN 11/24 UNDER APT WITH UNIBUS MEMORY AND NO MAIN MEMORY, AND BECAUSE OF NO CACHE IN AN 11/24, DESELECTS THE OPTIONAL CACHE TESTS.

1ST PASS	LONGEST	ADDITIONAL
RUN TIME	TEST TIME	RUN TIME
10	5	0

.....	E TABLES		
	A	B	C	D
E-MODE/S-MODE (\$ENVM/\$ENV)	000/000	240/001	240/001	240/001
SWITCH REGISTER 1 (\$SWREG)	000100	000100	000000	004000
SWITCH REGISTER 2	000000	000000	000000	000000
CPU TYPE/OPTIONS	00/0000	00/0000	00/0000	00/0000
MEMORY MAP CODE 1	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 2	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 3	000/0000	000/0000	000/0000	000/0000
MEMORY MAP CODE 4	000/0000	000/0000	000/0000	000/0000
BUS PRIORITY/INTERRUPT 1	0000	0000	0000	0000
BUS PRIORITY/INTERRUPT 2	0000	0000	0000	0000
BASE ADDRESS CODE	000000	000000	000000	000000
DEVICE MAP CODE	000000	000000	000000	000000
CTLR. SPECIFIC WORD 1	000000	000000	000000	000000
CTLR. SPECIFIC WORD 2	000000	000000	000000	000000
DEVICE DESCRIPTOR WORD 0	177777	177777	177777	177777
DEVICE DESCRIPTOR WORD 1	177777	177777	177777	177777
DEVICE DESCRIPTOR WORD 2	000000	000000	000000	000000
	THROUGH			
DEVICE DESCRIPTOR WORD 15	000000	000000	000000	000000

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
324
325
326
327
328
329
330
331
332
333
334

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

WHEN AN ERROR IS DETECTED AN 'ERROR' (EMT) INSTRUCTION IS EXECUTED AND THE 'ERROR HANDLER ROUTINE' CHECKS THE SWITCH REGISTER FOR MODE SELECTED.

THE PROGRAM WILL:

HALT ON ERROR	IF SW15=1
INHIBIT ERROR TYPE OUT	IF SW13=1
RING BELL ON ERROR	IF SW10=1
LOOP ON ERROR	IF SW9=1

6.2 ERROR RECOVERY

IF SW09=1, THE PROGRAM WILL LOOP BACK TO THE POINT WHERE THE INSTRUCTION THAT CAUSED THE ERROR WAS EXECUTED, WITHOUT ALLOWING ANY OF THE CONDITIONS TO CHANGE. THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP.

IF SW09=0, EACH ERROR WILL BE REPORTED AND LOGGED AND, AT THE END OF EACH TEST, A SUMMARY OF ALL ERRORS OCCURRING IN THAT TEST WILL BE PROVIDED. THE SUMMARY CONSISTS OF THE LOGICAL AND AND OR OF THE ADDRESS AND/OR DATA THAT WAS WRONG.

IF THE POWER MONITOR BIT ERROR IS CALLED, YOU MUST CORRECT THE POWER SUPPLY OR CPU ERROR REGISTER PROBLEM BEFORE YOU CAN RELY ON THE RESULTS OF THIS DIAGNOSTIC. THE ERROR CAN BE CALLED AT 1 OF 2 PLACES - 1) IN THE SCOPE ROUTINE EXECUTED AT THE BEGINNING OF EACH TEST, AND 2) IN AN ERROR CALL IN CASE BIT BECOMES SET AFTER THE SCOPE. IF THE BIT IS CAUGHT IN THE ERROR ROUTINE, *TWO* ERRORS WILL CALL - 1) POWER MONITOR BIT ERROR WILL CALL FIRST TO ALERT YOU TO THE POSSIBILITY THE PROBLEM COULD BE CAUSED BY THE OUT-OF-SPEC POWER SUPPLY, AND THEN THE ERROR THAT WAS TO CALL.

6.3 SAMPLE ERROR TYPE OUTS SEE '\$ERRTB:' FOR SAMPLE ERROR TYPEOUTS.

335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353

6.3.1 MULTIPLE TYPE ERRORS: AN EXAMPLE:

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

REG.ADR	TESTNO	ERRORPC
170210	000001	015226
170212	000001	015232
.	.	.
.	TO	.
.	.	.
170372	000001	015232
170374	000001	015232
170376	000001	015232

SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ

REGADRS	REGADRS	#ERRORS	TESTNO	ERRORPC
'OR'	'AND'			
170376	170210	32	000001	010530

354	7.	RESTRICTIONS
355		
356	7.1	STARTING RESTRICTIONS
357		
358		NONE
359		
360	7.2	OPERATING RESTRICTIONS
361		
362		NONE

363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE RUN TIME FOR ANY PASS IS APPROXIMATELY 3 SECONDS.

8.2 ADDRESS GENERATION IN THE PDP-11/44

THE FOLLOWING IS AN EXAMPLE OF HOW A MEMORY ADDRESS IS GENERATED BY THE UNIBUS MAP. THIS ASSUMES THAT THE ADDRESS ORIGINATES IN THE CPU BUT THE PROCESS CAN APPLY TO ANY UNIBUS ADDRESS, STARTING AT LINE C2.

A. VIRTUAL ADDRESS	15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
A1. P.A.R. PAGE NUMBER (0-7)	15 14 13
A2. OFFSET (FROM VIRTUAL ADDRESS)	12 11 10 09 08 07 06 05 04 03 02 01 00
B. P.A.R.[PAGE NO.] +	15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
C. PHYS ADDRS (A2+B)	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
C1. 17XXXXXX=> U.B.ADR.	21 20 19 18
C2. MAPPING REG.NO.(0-36)	17 16 15 14 13
C3. OFFSET	12 11 10 09 08 07 06 05 04 03 02 01 00
D. MAP REG.[NO.] +	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01
E. PHYS ADDR (C3+D)	21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)

A1: UPPER 3 BITS OF VIRTUAL ADDRESS, USED TO SELECT A PAGE ADDRESS REGISTER (PAR)

A2: LOWER 13 BITS OF VIRTUAL ADDRESS, ADDED TO SELECTED PAR

B: PAGE ADDRESS REGISTER (16 BITS), IN ADDITION PROCESS THIS GETS LEFT SHIFTED 6 BITS BEFORE ADDITION TO A2

C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT, (22 BITS)

C1: IF UPPER 4 BITS ARE ALL ONES THEN BITS <17:00> GO OUT ON UNIBUS

C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELECT ONE OF THE 36 (OCTAL) MAP REGISTERS.

C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED MAP REGISTER

D: MAP REGISTER (22 BITS), ADDED TO BITS <12:00> OF UNIBUS ADDRESS

E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACHE.

417
418
419
420
421

9. PROGRAM DESCRIPTION

THE ASSEMBLED LISTING, CKKUAE.SEQ, HAS A PARAGRAPH DESCRIBING EACH OF THE TESTS. THE PARAGRAPH WILL INDICATE IF THE TEST IS RUN CONDITIONALLY ON THE STATUS ON THE CACHE CONTROL REGISTER.@

1485

```

.TITLE CKKUAE0 11/24/44 UBI MAP
.*COPYRIGHT (C) JANUARY 1982
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY DAN P. MILLEVILLE
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.

```

1486

```

.SBTTL OPERATIONAL SWITCH SETTINGS

```

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT TRACE TRAP
11	INHIBIT TRACE TRAP
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<4:0>
7	INHIBIT MULTIPLE ERROR TYPEOUTS
6	SELECT CACHE-CIS TESTS
5	SELECT MEMORY ON UNIBUS TEST

1487

1488

1489

```

.SBTTL BASIC DEFINITIONS
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100                ;;FIRST ADDRESS OF THE STACK
001100 KERSTK= STACK           ;;KERNEL STACK
000700 SUPSTK= STACK-200      ;;SUPERVISOR STACK
000600 USESTK= STACK-300      ;;USER STACK
104000 ERROR=EMT
000004 SCOPE=IOT
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
177546 LKS= 177546         ;;LINE CLOCK (KW11-L) STATUS REGISTER
:*MISCELLANEOUS DEFINITIONS
000011 HT= 11                ;;CODE FOR HORIZONTAL TAB
000012 LF= 12                ;;CODE LINE FEED
000015 CR= 15                ;;CODE CARRIAGE RETURN
000200 CRLF= 200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
:*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
000000 R10=R0
000001 R11=R1
000002 R12=R2
000003 R13=R3
000004 R14=R4
000005 R15=R5
000006 SP= %6                ;;STACK POINTER
000006 KSP=SP
000006 SSP=SP
000006 USP=SP
000007 PC= %7                ;;PROGRAM COUNTER
:*PRIORITY LEVEL DEFINITIONS
000000 PR0= 0                ;;PRIORITY LEVEL 0
000040 PR1= 40               ;;PRIORITY LEVEL 1
000100 PR2= 100              ;;PRIORITY LEVEL 2
000140 PR3= 140              ;;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
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1

SW9=SW09
SW8=SW08
SW7=SW07
SW6=SW06
SW5=SW05
SW4=SW04
SW3=SW03
SW2=SW02
SW1=SW01
SW0=SW00

.....

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

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

BIT9=BIT09
BIT8=BIT08
BIT7=BIT07
BIT6=BIT06
BIT5=BIT05
BIT4=BIT04
BIT3=BIT03
BIT2=BIT02
BIT1=BIT01
BIT0=BIT00

.....

;*BASIC "CPU" TRAP VECTOR ADDRESSES

000004
000010
000014
000014
000014
000020
000024
000030
000034
000060

ERRVEC= 4
RESVEC= 10
TBITVEC=14
TRTVEC= 14
BPTVEC= 14
IOTVEC= 20
PWRVEC= 24
EMTVEC= 30
TRAPVEC=34
TKVEC= 60

:::TIME OUT AND OTHER ERRORS
:::RESERVED AND ILLEGAL INSTRUCTIONS
:::'T' BIT
:::TRACE TRAP
:::BREAKPOINT TRAP (BPT)
:::INPUT/OUTPUT TRAP (IOT) **SCOPE**
:::POWER FAIL
:::EMULATOR TRAP (EMT) **ERROR**
:::'TRAP' TRAP
:::TTY KEYBOARD VECTOR

```

000064 TPVEC= 64          ;;TTY PRINTER VECTOR
000100 LKVEC= 100         ;;LINE CLOCK (KW11-L) VECTER
000114 CACHVEC=114      ;;CACHE ERROR INTERRUPT VECTOR
000240 PIRQVEC=240      ;;PROGRAM INTERRUPT REQUEST VECTOR
000250 MMVEC= 250        ;;MEMORY MANAGEMENT VECTOR

.SBTTL CACHE REGISTER DEFINITIONS
177740 LOADRS = 177740 ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
177742 HIADRS = 177742 ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
177744 MEMERR = 177744 ;;CACHE ERROR REGISTER
177746 CONTRL = 177746 ;;MEMORY CONTROL REGISTER
177750 MAINT = 177750 ;;MEMORY MAINTENANCE REGISTER
177752 HITMIS = 177752 ;;HIT MISS REGISTER '1' IMPLIES HIT IN CACHE

.SBTTL CPU REGISTER DEFINITIONS
177760 SIZELO = 177760 ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
;;TO GET TO THE LAST 32 WORDS OF MEMORY
177762 SIZEHI = 177762 ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
;;CURRENTLY ALL ZERO
177764 SYSTID = 177764 ;;SYSTEM ID REGISTER
177766 CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
;;THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS
;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572 MMR0= 177572
177574 MMR1= 177574
177576 MMR2= 177576
172516 MMR3= 172516
177572 SR0=MMR0
177574 SR1=MMR1
177576 SR2=MMR2
172516 SR3=MMR3

;*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
177650 UIPAR4= 177650
177652 UIPAR5= 177652
177654 UIPAR6= 177654
177656 UIPAR7= 177656
    
```

```

177660      ;*USER 'D' PAGE ADDRESS REGISTERS
177662      UDPAR0= 177660
177664      UDPAR1= 177662
177666      UDPAR2= 177664
177670      UDPAR3= 177666
177672      UDPAR4= 177670
177674      UDPAR5= 177672
177676      UDPAR6= 177674
177676      UDPAR7= 177676
172200      ;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
172202      SIPDR0= 172200
172204      SIPDR1= 172202
172206      SIPDR2= 172204
172210      SIPDR3= 172206
172212      SIPDR4= 172210
172214      SIPDR5= 172212
172216      SIPDR6= 172214
172216      SIPDR7= 172216
172220      ;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
172222      SDPDR0= 172220
172224      SDPDR1= 172222
172226      SDPDR2= 172224
172230      SDPDR3= 172226
172232      SDPDR4= 172230
172234      SDPDR5= 172232
172236      SDPDR6= 172234
172236      SDPDR7= 172236
172240      ;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
172242      SIPAR0= 172240
172244      SIPAR1= 172242
172246      SIPAR2= 172244
172250      SIPAR3= 172246
172252      SIPAR4= 172250
172254      SIPAR5= 172252
172256      SIPAR6= 172254
172256      SIPAR7= 172256
172260      ;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
172262      SDPAR0= 172260
172264      SDPAR1= 172262
172266      SDPAR2= 172264
172270      SDPAR3= 172266
172272      SDPAR4= 172270
172274      SDPAR5= 172272
172276      SDPAR6= 172274
172276      SDPAR7= 172276
172300      ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
172302      KIPDR0= 172300
172304      KIPDR1= 172302
172306      KIPDR2= 172304
172310      KIPDR3= 172306
172312      KIPDR4= 172310
172314      KIPDR5= 172312
172316      KIPDR6= 172314
172316      KIPDR7= 172316
172320      ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
172322      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 UNIBUS MAP REGISTER DEFINITIONS
;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200 MAPL00 = 170200
170202 MAPH00 = 170202
170204 MAPL01 = 170204
170206 MAPH01 = 170206
170210 MAPL02 = 170210
170212 MAPH02 = 170212
170214 MAPL03 = 170214
170216 MAPH03 = 170216
170220 MAPL04 = 170220
170222 MAPH04 = 170222
170224 MAPL05 = 170224
170226 MAPH05 = 170226
170230 MAPL06 = 170230
170232 MAPH06 = 170232
170234 MAPL07 = 170234
170236 MAPH07 = 170236
170240 MAPL10 = 170240
170242 MAPH10 = 170242
170244 MAPL11 = 170244
170246 MAPH11 = 170246
170250 MAPL12 = 170250
170252 MAPH12 = 170252
170254 MAPL13 = 170254
170256 MAPH13 = 170256
170260 MAPL14 = 170260
170262 MAPH14 = 170262
170264 MAPL15 = 170264
170266 MAPH15 = 170266
170270 MAPL16 = 170270
170272 MAPH16 = 170272
```

170274	MAPL17 = 170274
170276	MAPH17 = 170276
170300	MAPL20 = 170300
170302	MAPH20 = 170302
170304	MAPL21 = 170304
170306	MAPH21 = 170306
170310	MAPL22 = 170310
170312	MAPH22 = 170312
170314	MAPL23 = 170314
170316	MAPH23 = 170316
170320	MAPL24 = 170320
170320	MAPH24 = 170320
170324	MAPL25 = 170324
170326	MAPH25 = 170326
170330	MAPL26 = 170330
170332	MAPH26 = 170332
170334	MAPL27 = 170334
170336	MAPH27 = 170336
170340	MAPL30 = 170340
170342	MAPH30 = 170342
170344	MAPL31 = 170344
170346	MAPH31 = 170346
170350	MAPL32 = 170350
170352	MAPH32 = 170352
170354	MAPL33 = 170354
170356	MAPH33 = 170356
170360	MAPL34 = 170360
170362	MAPH34 = 170362
170364	MAPL35 = 170364
170366	MAPH35 = 170366
170370	MAPL36 = 170370
170372	MAPH36 = 170372
170374	MAPL37 = 170374
170376	MAPH37 = 170376
170200	MAPL0=MAPL00
170202	MAPH0=MAPH00
170204	MAPL1=MAPL01
170206	MAPH1=MAPH01
170210	MAPL2=MAPL02
170212	MAPH2=MAPH02
170214	MAPL3=MAPL03
170216	MAPH3=MAPH03
170220	MAPL4=MAPL04
170222	MAPH4=MAPH04
170224	MAPL5=MAPL05
170226	MAPH5=MAPH05
170230	MAPL6=MAPL06
170232	MAPH6=MAPH06
170234	MAPL7=MAPL07
170236	MAPH7=MAPH07

.....

1492
000000
000174 000174
000174 000000
000176 000000
000200 000137 010000

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

1494

000204
000046
000046 022024
000052 000052
000052 000000
000204 000204

```
.SBTTL ACT11 HOOKS  
:*****  
:HOOKS REQUIRED BY ACT11  
    $SVPC=.           ;SAVE PC  
    .=46              ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP  
    $ENDAD            ;;2)SET LOC.52 TO ZERO  
    .=52              ;; RESTORE PC  
    .WORD 0  
    .=$SVPC
```


1496

.SBTTL COMMON TAGS

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

001100	001100	\$CMTAG:	.=1100	::START OF COMMON TAGS
001100	000	:\$PASS:	.WORD 0	::CONTAINS PASS COUNT ;DPM001
001101	000	\$TSTNM:	.BYTE 0	::CONTAINS THE TEST NUMBER
001102	000000	\$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	000	\$ERTTL:	.WORD 0	::CONTAINS TOTAL ERRORS DETECTED
001113	001	\$ITEMB:	.BYTE 0	::CONTAINS ITEM CONTROL BYTE
001114	000000	\$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	000	\$AUTOB:	.BYTE 0	::AUTOMATIC MODE INDICATOR
001133	000	\$INTAG:	.BYTE 0	::INTERRUPT MODE INDICATOR
001134	000000		.WORD 0	
001136	177570	\$SWR:	.WORD DSWR	::ADDRESS OF SWITCH REGISTER
001140	177570	\$DISPLAY:	.WORD DDISP	::ADDRESS OF DISPLAY REGISTER
001142	177560	\$TKS:	177560	::TTY KBD STATUS
001144	177562	\$TKB:	177562	::TTY KBD BUFFER
001146	177564	\$TPS:	177564	::TTY PRINTER STATUS REG. ADDRESS
001150	177566	\$TPB:	177566	::TTY PRINTER BUFFER REG. ADDRESS
001152	000	\$NULL:	.BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
001153	002	\$FILLS:	.BYTE 2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001154	012	\$FILLC:	.BYTE 12	::INSERT FILL CHARS. AFTER A "LINE FEED"
001155	000	\$TPFLG:	.BYTE 0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001156	000000	\$REGAD:	.WORD 0	::CONTAINS THE ADDRESS FROM WHICH (\$REG0) WAS OBTAINED
001160	000006	\$REG0:	.REPT .WORD 0	::CONTAINS ((\$REGAD)+0)
001162	000000	\$REG1:	.WORD 0	::CONTAINS ((\$REGAD)+2)
001164	000000	\$REG2:	.WORD 0	::CONTAINS ((\$REGAD)+4)
001166	000000	\$REG3:	.WORD 0	::CONTAINS ((\$REGAD)+6)
001170	000000	\$REG4:	.WORD 0	::CONTAINS ((\$REGAD)+10)
001172	000000	\$REG5:	.WORD 0	::CONTAINS ((\$REGAD)+12)
001174	000000	\$TMP0:	.REPT .WORD 0	::USER DEFINED
001176	000000	\$TMP1:	.WORD 0	::USER DEFINED
001200	000000	\$TMP2:	.WORD 0	::USER DEFINED
001202	000000	\$TMP3:	.WORD 0	::USER DEFINED
001204	000000	\$TMP4:	.WORD 0	::USER DEFINED
001206	000000	\$TMP5:	.WORD 0	::USER DEFINED
001210	000000	\$TMP6:	.WORD 0	::USER DEFINED
001212	000000	\$TIMES:	0	::MAX. NUMBER OF ITERATIONS
001214	000000	\$ESCAPE:	0	::ESCAPE ON ERROR ADDRESS
001216	207	\$BELL:	.ASCIZ <207><377><377>	::CODE FOR BELL

377

377

001222	077		\$QUES: .ASCII	/?/	::QUESTION MARK
001223	015		\$CRLF: .ASCII	<15>	::CARRIAGE RETURN
001224	012	000	\$LF: .ASCIZ	<12>	::LINE FEED
:*****					
001226	000000		PADRSL: .WORD	0	:HOLDS THE LOWER 16 BITS OF A 22 BIT ADDRESS :GENERATED FOR TYPE OUT.
001230	000000		PADRSH: .WORD	0	:HOLDS THE UPPER 6 BITS OF A 22 BIT ADDRESS :GENERATED FOR TYPE OUT.
001232	000000	000077	ADRAND: .WORD	0,77	:LOGICAL AND OF FAILING ADDRESSES
001236	000000	000077	ADDROR: .WORD	0,77	:LOGICAL OR OF FAILING ADDRESSES
001242	000000	000077	DATAND: .WORD	0,77	:LOGICAL AND OF BAD DATA
001246	000000	000077	DATAOR: .WORD	0,77	:LOGICAL OR OF BAD DATA
001252	000000		PATAND: .WORD	0	:LOGICAL AND OF PATTERN LOADED
001254	000000		PATOR: .WORD	0	:LOGICAL OR OF PATTERN LOADED
001256	000000		LOWEST: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE :LOWEST USABLE MAP REGISTER TO RESPOND
001260	000000		HIGEST: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO CAUSE THE :HIGHEST USABLE MAP REGISTER TO RESPOND
001262	000000		UBMLOW: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO SIGNAL 1ST :ADDRESS OF UNIBUS MEMORY
001264	000000		UBMHI: .WORD	0	:HOLDS NUMBER TO PUT IN PAR TO SIGNAL LAST :BLOCK OF 4K OF UNIBUS MEMORY
001266	000000		MMRLOW: .WORD	0	:HOLDS LOWEST MAP REGISTER NUMBER FROM 'LOWEST:'
001270	000000		MMRHI: .WORD	0	:HOLDS HIGHEST MAP REGISTER NUMBER FROM 'HIGEST:'
001272	000000		UBRLOW: .WORD	0	:HOLDS LOWEST MAP REGISTER NUMBER FROM 'UBMLOW:'
001274	000000		UBRHI: .WORD	0	:HOLDS HIGHEST MAP REGISTER NUMBER FROM 'UBMHI:'
001276	000000		BUPWIN: .WORD	0	:HOLDS LOWEST USEABLE PAR OF UPPER WINDOW
001300	000000		LREGL: .WORD	0	:HOLDS I/O PAGE ADDR OF LOW 16 BITS OF :THE LOWEST USABLE MAP REGISTER
001302	000000		LREGU: .WORD	0	:HOLDS I/O PAGE ADDR OF HIGH 6 BITS OF :OF THE LOWEST USABLE MAP REGISTER
001304	000000		LMAH: .WORD	0	:LOCATION TO HOLD LMA HIGH REGISTER CONTENTS
001306	000000		LMAL: .WORD	0	:LOCATION TO HOLD LMA LOW REGISTER CONTENTS
001310	000000		UBM24L: .WORD	0	:LOCATION USED TO HOLD LMA LOW EXPECTED VALUE
001312	000000		UBM24U: .WORD	0	:LOCATION USED TO HOLD LMA HIGH EXPECTED VALUE
001314	000000		UBM24P: .WORD	0	:LOCATION USED TO HOLD LMA LOW PRELOAD VALUE
001316	000000		NUMOFK: .WORD	0	:LOCATION TO HOLD NUMBER OF K OF UB MEMORY
001320	000000		ERRCNT: .WORD	0	:MULTIPLE ERROR ERROR COUNTER
001322	000000		CNTR: .WORD	0	:AUXILIARY COUNTER
001324	000000		FLAG: .WORD	0	:FLAG TO INDICATE TO LAST PROGRAM PASS N
001326	000000		CPUEXP: .WORD	0	:HOLDS THE EXPECTED CPU ERROR CODE
001330	000000		PCPUER: .WORD	0	:HOLDS RECEIVED CPU ERROR CONDITION
001332	000000		PPARER: .WORD	0	:HOLDS RECEIVED PARITY ERROR CONDITION
001334	000000		PCONTR: .WORD	0	:HOLDS CONTENTS OF CONTROL REGISTER
001336	000000		PMAINT: .WORD	0	:HOLDS CONTENTS OF MAINTENANCE REGISTER
001340	000000		BADPC: .WORD	0	:HOLDS PC OF INST THAT CAUSED TRAP
001342	000000		OLDPC: .WORD	0	:HOLDS THE RETURN ADDRESS AFTER A TRAP
001344	000000		OLDPS: .WORD	0	:HOLDS THE OLD PROCESSOR STATUS
001346	000000		OLDPSW: .WORD	0	:HOLDS OLD PSW FOR TBITRESTORE
001350	000000		PMMR0: .WORD	0	:HOLDS CONTENTS OF MMR0 AFTER TRAP
001352	000000		PMMR1: .WORD	0	:HOLDS CONTENTS OF MMR1 AFTER TRAP
001354	000000		PMMR2: .WORD	0	:HOLDS CONTENTS OF MMR2 AFTER TRAP
001356	000000		RSIZE: .WORD	0	:WILL HOLD P.A.R. DATA FOR TOP OF MEMORY
001360	000000		RETRY: .WORD	0	:RETRY FLAG IN CASE OF PARITY ABORTS
001362	000000		NXTTST: .WORD	0	:LOCATION TO HOLD ESCAPE ADDRESS ON :PARITY ERRORS.
001364	000200		DATA: .WORD	200	:PATTERN TO BE USED TO LOAD INTO MEMORY

.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

1497	001366				
1498	001366	024160	.WORD	EM1	:NOT THE CORRECT CPU TRAP CONDITION THROUGH ERRVEC (#004)
1499	001370	030121	.WORD	DH1	:RECEIVD EXPECTD TESTNO PC AT ABORT
1500	001372	032340	.WORD	DT1	:PCPUER,CPUEXP,\$TESTN,BADPC,0
1501	001374	033641	.WORD	DF1	: 0, 0, 0, 0
1502					
1503					
1504	001376	024245	.WORD	EM2	:UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)
1505	001400	030160	.WORD	DH2	:RECEIVD TESTNO PC AT ABORT
1506	001402	032352	.WORD	DT2	:PCPUER,\$TESTN,BADPC,0
1507	001404	033641	.WORD	DF1	: 0, 0, 0
1508					
1509					
1510	001406	024317	.WORD	EM3	:MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS
1511	001410	030207	.WORD	DH3	:STATUS AUTOI/D VIRTADR
1512					:REGISTR REGISTR REGISTR TESTNO PC AT ABORT
1513	001412	032362	.WORD	DT3	:PMMR0,PMMR1,PMMR2,\$TESTN,BADPC,0
1514	001414	033641	.WORD	DF1	: 0, 0, 0, 0, 0
1515					
1516					
1517	001416	024425	.WORD	EM4	:SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
1518	001420	030306	.WORD	DH4	:REGADRS REGADRS
1519					: "OR" "AND" #ERRORS TESTNO ERR PC
1520	001422	032376	.WORD	DT4	:ADDROR,ADRAND,ERRCNT,\$TESTN,\$ERRPC,0
1521	001424	033646	.WORD	DF4	: 2, 2, 1, 0, 0
1522					
1523					
1524	001426	024505	.WORD	EM5	:SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS
1525	001430	030403	.WORD	DH5	:REGLOAD REGLOAD REGDUAL REGDUAL
1526					: "OR" "AND" "OR" "AND" #ERRORS TESTNO
1527	001432	032412	.WORD	DT5	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
1528	001434	033653	.WORD	DF5	: 2, 2, 2, 2, 1, 0
1529					
1530					
1531	001436	024600	.WORD	EM6	:SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS
1532	001440	030540	.WORD	DH6	:MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD
1533					: "OR" "AND" "OR" "AND" "OR" "AND" #ERRORS TESTNO
1534	001442	032430	.WORD	DT6	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
1535	001444	033661	.WORD	DF6	: 2, 2, 0, 0, 0, 0, 1, 0

1536			:ITEM 7					
1537	001446	024702		.WORD	EM7	:SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS		
1538	001450	030540		.WORD	DH6	:MAPREG	MAPREG	EXPECTD
1539						: 'OR'	'AND'	'OR'
1540	001452	032430		.WORD	DT6	:ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,\$TESTN,0		
1541	001454	033661		.WORD	DF6	: 2, 2, 0, 0, 0, 0, 1, 0		
1542								
1543			:ITEM 10					
1544	001456	025003		.WORD	EM10	:CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF		
1545						:SO JUMPING TO THE SIZE JUMPER TEST FOR VERIFICATION		
1546	001460	030727		.WORD	DH10	:TESTNO	ERR PC	
1547	001462	032452		.WORD	DT10	: \$TESTN,\$ERRPC,0		
1548	001464	033641		.WORD	DF1	: 0, 0		
1549								
1550			:ITEM 11					
1551	001466	025155		.WORD	EM11	:SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH		
1552	001470	030754		.WORD	DH11	:EXPECTD	EXPECTD	RECEIVD
1553						: 'OR'	'AND'	'OR'
1554	001472	032462		.WORD	DT11	:PATTOR,PATAND,DATAOR,DATAND,ERRCTN,\$TESTN,0		
1555	001474	033671		.WORD	DF11	: 0, 0, 0, 0, 1, 0		
1556								
1557			:ITEM 12					
1558	001476	025247		.WORD	EM12	:UNIBUS MAP IS RELOCATING WHEN NOT ENABLED		
1559	001500	030727		.WORD	DH10	:TESTNO	ERR PC	
1560	001502	032452		.WORD	DT10	: \$TESTN,\$ERRPC,0		
1561	001504	033641		.WORD	DF1	: 0, 0		
1562								
1563			:ITEM 13					
1564	001506	025321		.WORD	EM13	:CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL		
1565						:ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM		
1566						:IF YOU DON'T LOOP ON THIS PROBLEM.		
1567	001510	030727		.WORD	DH10	:TESTNO	ERR PC	
1568	001512	032452		.WORD	DT10	: \$TESTN,\$ERRPC,0		
1569	001514	033641		.WORD	DF1	: 0, 0		
1570								
1571			:ITEM 14					
1572	001516	025525		.WORD	EM14	:SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED		
1573	001520	030754		.WORD	DH11	:EXPECTD	EXPECTD	RECEIVD
1574						: 'OR'	'AND'	'OR'
1575	001522	032500		.WORD	DT14	:ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,\$TESTN,0		
1576	001524	033677		.WORD	DF14	: 2, 2, 0, 0, 1, 0		
1577								
1578			:ITEM 15					
1579	001526	025630		.WORD	EM15	:MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.		
1580	001530	031111		.WORD	DH15	:CONDITN	CONDITN	
1581						:EXPECTD	RECEIVD	TESTNO
1582	001532	032516		.WORD	DT15	:CPUEXP,PCPUER,\$TESTN,\$ERRPC,0		
1583	001534	033641		.WORD	DF1	: 0, 0, 0, 0		
1584								
1585			:ITEM 16					
1586	001536	025723		.WORD	EM16	:SUMMARY OF DUAL MAPPING ERRORS		
1587	001540	030754		.WORD	DH11	:EXPECTD	EXPECTD	RECEIVD
1588						: 'OR'	'AND'	'OR'
1589	001542	032500		.WORD	DT14	:ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,\$TESTN,0		
1590	001544	033653		.WORD	DF5	: 2, 2, 2, 2, 1, 0		

1591			:ITEM 17		
1592	001546	026021	.WORD	EM17	:NO UNIBUS MEMORY EXISTS
1593	001550	030727	.WORD	DH10	:TESTNO ERR PC
1594	001552	032452	.WORD	DT10	:\$TESTN,\$ERRPC,0
1595	001554	033641	.WORD	DF1	: 0, 0
1596					
1597			:ITEM 20		
1598	001556	026051	.WORD	EM20	:INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR
1599	001560	030727	.WORD	DH10	:TESTNO ERR PC
1600	001562	032452	.WORD	DT10	:\$TESTN,\$ERRPC,0
1601	001564	033641	.WORD	DF1	: 0, 0
1602					
1603			:ITEM 21		
1604	001566	026150	.WORD	EM21	:INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
1605					:DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
1606	001570	030727	.WORD	DH10	:TESTNO ERR PC
1607	001572	032452	.WORD	DT10	:\$TESTN,\$ERRPC,0
1608	001574	033641	.WORD	DF1	: 0, 0
1609					
1610			:ITEM 22		
1611	001576	026226	.WORD	EM22	:INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
1612	001600	030727	.WORD	DH10	:TESTNO ERR PC
1613	001602	032452	.WORD	DT10	:\$TESTN,\$ERRPC,0
1614	001604	033641	.WORD	DF1	: 0, 0
1615					
1616			:ITEM 23		
1617	001606	026314	.WORD	EM23	:LMA NOT LOADED PROPERLY
1618	001610	031170	.WORD	DH23	:TESTNO ERR PC LMAEXP LMARCV
1619	001612	032530	.WORD	DT23	:\$TESTN,\$ERRPC,EADRES,EADRS2,0
1620	001614	033705	.WORD	DF23	: 0, 0, 2, 2
1621					
1622			:ITEM 24		
1623	001616	026344	.WORD	EM24	:LMA FORCE JUMPER BIT NOT ZERO
1624	001620	031231	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
1625	001622	032542	.WORD	DT24	:\$TESTN,\$ERRPC,\$REG1,LMAHI,0
1626	001624	033641	.WORD	DF1	: 0, 0, 0, 0
1627					
1628			:ITEM 25		
1629	001626	026402	.WORD	EM25	:LMA FORCE JUMPER BIT NOT SET
1630	001630	031231	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
1631	001632	032542	.WORD	DT24	:\$TESTN,\$ERRPC,\$REG1,LMAHI,0
1632	001634	033641	.WORD	DF1	: 0, 0, 0, 0
1633					
1634			:ITEM 26		
1635	001636	026437	.WORD	EM26	:LMA CONTROL BITS INCORRECT
1636	001640	031231	.WORD	DH24	:TESTNO ERR PC LMAEXP LMARCV
1637	001642	032554	.WORD	DT26	:\$TESTN,\$ERRPC,\$TMP0,\$REG2,0
1638	001644	033641	.WORD	DF1	: 0, 0, 0, 0
1639					
1640			:ITEM 27		
1641	001646	026472	.WORD	EM27	:FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT
1642	001650	031270	.WORD	DH27	:TESTNO ERR PC LMARCV KIPAR4
1643	001652	032566	.WORD	DT27	:\$TESTN,\$ERRPC,\$TMP0,KIPAR4,0
1644	001654	033641	.WORD	DF1	: 0, 0, 0

1645
1646 001656 026572
1647 001660 031327
1648 001662 032600
1649 001664 033641

;ITEM 30

.WORD EM30 ;KIPAR5 NOT LOADED PROPERLY
.WORD DH30 ;TESTNO ERR PC PR5EXP PR5RCV
.WORD DT30 ;\$TESTN,\$ERRPC,\$TMP5,KIPAR5,0
.WORD DF1 ; 0, 0, 0, 0

```

1650 001666
1651
1652
1653
1654
1655 001666 026625
1656 001670 031366
1657 001672 032612
1658 001674 033711
1659
1660
1661 001676 026701
1662 001700 031415
1663
1664 001702 032622
1665 001704 033661
1666
1667
1668 001706 026774
1669 001710 031524
1670 001712 032636
1671 001714 033711
1672
1673
1674 001716 027055
1675 001720 031605
1676 001722 032654
1677 001724 033717
1678
1679
1680 001726 027124
1681 001730 031656
1682
1683 001732 032670
1684 001734 033677
1685
1686
1687 001736 027206
1688 001740 031743
1689 001742 032702
1690 001744 033711
1691
1692
1693 001746 027237
1694 001750 032013
1695 001752 032716
1696 001754 033711
1697
1698
1699 001756 027346
1700 001760 032054
1701 001762 032730
1702 001764 033724

```

```

ER200:
;THIS IS THE STARTING POINT FOR ERROR MESSAGES
;201 THROUGH 377. THEY ARE USED FOR MULTIPLE
;ERROR MESSAGES.

;ITEM 201
.WORD EM201 ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
.WORD DH201 ;REGADRS TESTNO ERR PC
.WORD DT201 ;EADRES,$TESTN,$ERRPC,0
.WORD DF201 ; 2, 0, 0

;ITEM 202
.WORD EM202 ;THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP
.WORD DH202 ;MAPREG MAPREG NON-ZER
;TESTING DUALED CNTNTS TESTNO ERR PC
.WORD DT202 ;EADRES,EADRS2,$TMP3,$TESTN,$ERRPC,0
.WORD DF6 ; 2, 2, 0, 0, 0

;ITEM 203
.WORD EM203 ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
.WORD DH203 ;REGADRS PATRN EXPCTD RECEVD TESTNO ERR PC
.WORD DT203 ;EADRS2,$TMP0,$REG4,$REG3,$TESTN,$ERRPC,0
.WORD DF201 ; 2, 0, 0, 0, 0, 0

;ITEM 204
.WORD EM204 ;UNIBUS DATA PATH COUNT PATTERN FAILURE
.WORD DH204 ;EXPECTD RECEVD ADDRLOAD TESTNO ERR PC
.WORD DT204 ;$TMP0,$TMP1,$REG2,$TESTN,$ERRPC,0
.WORD DF204 ;0, 0, 3, 0, 0

;ITEM 205
.WORD EM205 ;UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
.WORD DH205 ;ADDRESS ADDRESS
;EXPECTD RECEVD TESTNO ERR PC
.WORD DT205 ;EADRES,EADRS2,$TESTN,$ERRPC,0
.WORD DF14 ; 2, 2, 0, 0

;ITEM 206
.WORD EM206 ;DATA PATTERN NOT CORRECT
.WORD DH206 ;ADDRESS EXPCTD RECVD TESTNO ERR PC
.WORD DT206 ;EADRES,$TMP4,$TMP5,$TESTN,$ERRPC,2
.WORD DF201 ; 2, 0, 0, 0, 0

;ITEM 207
.WORD EM207 ;REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770
.WORD DH207 ;ADDRUSED BITDIFF TESTNO ERR PC
.WORD DT207 ;EADRES,$REG0,$TESTN,$ERRPC,0
.WORD DF201 ;2, 0, 0, 0

;ITEM 210
.WORD EM210 ;MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST
.WORD DH210 ;TESTNO ERR PC MAPREGADR
.WORD DT210 ;$TESTN,$ERRPC,EADRES,0
.WORD DF210 ; 0, 0, 2

```

1703					
1704	001766	027446			
1705					
1706	001770	032106			
1707					
1708	001772	032740			
1709	001774	033711			
1710					
1711					
1712	001776	027604			
1713					
1714	002000	032106			
1715					
1716	002002	032740			
1717	002004	033711			
1718					
1719					
1720	002006	027737			
1721	002010	032270			
1722	002012	032766			
1723	002014	033641			
1724					
1725					
1726	002016	030030			
1727	002020	032270			
1728	002022	032766			
1729	002024	033641			

			:ITEM 211		
			.WORD	EM211	:RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION
					:TEST BEING RUN OVER UNIBUS
			.WORD	DH211	:CORRECT EXPECTD RECEIVD
					:ADDRESS DATA FROM UB TESTNO ERR PC
			.WORD	DT211	:EADRES,\$REG3,\$REG2,\$TESTN,\$ERRPC,0
			.WORD	DF201	: 2, 0, 0, 0, 0
			:ITEM 212		
			.WORD	EM212	:MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY.
					:TEST BEING RUN OVER UNIBUS
			.WORD	DH211	:CONDITN CONDITN
					:EXPECTD RECEIVD TESTNO ERR PC
			.WORD	DT211	:CPUEXP,PCPUER,\$TESTN,\$ERRPC,0
			.WORD	DF201	: 0, 0, 0, 0
			:ITEM 213		
			.WORD	EM213	:MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED
			.WORD	DH213	:TESTNO ERR PC REG NO DDWDAT DDWADR
			.WORD	DT213	:\$TESTN,\$ERRPC,\$TMP0,\$TMP1,\$REG5,0
			.WORD	DF1	: 0, 0, 0, 0, 0
			:ITEM 214		
			.WORD	EM214	:MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED
			.WORD	DH213	:TESTNO ERR PC REG NO DDWDAT DDWADR
			.WORD	DT213	:\$TESTN,\$ERRPC,\$TMP0,\$TMP1,\$REG5,0
			.WORD	DF1	: 0, 0, 0, 0, 0

1730
1731 002026
1732 000176
1733 000176 000000
1734 002026

.SBTTL SOFTWARE SWITCH REGISTER LOCATION
 .\$Y= :SAVE ADDRESS LOCATION
 .=176 :ADDRESS TO SOFTWARE SWITCH REGISTER LOCATION
\$SSWR: .WORD 0 :LOCATION FOR SOFTWARE SWITCH REGISTER
 .=.\$Y :RETURN TO PREVIOUS ADDRESS LOCATION

1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777 002026 010046
 1778 002030 005000
 1779 002032 113700 001112
 1780 002036 001004
 1781 002040 013746 001114
 1782 002044 104402
 1783 002046 000565
 1784 002050 122700 000177
 1785 002054 001003
 1786 002056 012700 002432
 1787 002062 000451
 1788 002064 005300
 1789 002066 072027 000003
 1790 002072 100041
 1791 002074 023727 001320 000020

```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE
*****
*
* THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
* THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
* AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
* TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FOR EACH
* ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
* MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
* HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
* DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
* OF THE DATA TO BE TYPED. THE FORMAT OF THIS DATA IS
* CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
* THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
* THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
*
* 0 -16 BIT OCTAL FORMAT
* 1 -DECIMAL FORMAT
* 2 -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
* PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
* 3 -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
* ADDRESS IN KERNEL I-SPACE.
* 4 -18 BIT OCTAL FORMAT. DATA IS A 16 BIT NUMBER THAT
* WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
* SHIFTING IT 6 BITS.
* 5 -16 BIT OCTAL, SUPPRESS LEADING ZEROS
* 6 -16 BIT DECIMAL, SUPPRESS SPACES
* IF YOU SHOULD HAVE A NEED TO JUST TYPE A STRING OF
* NUMBERS, SET UP YOUR CODE THIS WAY:
* MOV #CONTINUE,-(SP) ;MOVE THE ADDRESS OF THE INSTRUCTION AFTER THE
* ;JUMP TO THE STACK
* MOV R0,-(SP) ;SAVE R0
* MOV R1,-(SP) ;AND R1 ON THE STACK
* MOV DTNAME,R0 ;MOVE THE ADDRESS OF THE DATA TABLE TO R0
* JMP TYPDAT ;SUBROUTINE IDENTIFIED IN CENTER OF THIS ROUTINE
* CONTINUE: NEXT INSTRUCTION
* AT A CONVENIENT SPOT, ALLOCATE THE FOLLOWING:
* *DTNAME: .WORD DTLIST,DFNAME ;IDENTIFY THE LIST NAME AND DATA FORMAT BELOW
* *DFNAME: .BYTE N,N,N,N,ETC. ;CONSTRUCT YOUR OWN DATA FORMAT LINE
* .EVEN
* *DTLIST: VAR1,VAR2,VAR3,VAR4,.....,$CRLF,0 ;VARIABLES YOU WANT TYPED
*
*****
ERTYPE: MOV R0,-(KSP) ;SAVE R0 ON STACK
CLR R0 ;CLEAR R0
MOVB $ITEMB,R0 ;PUT ITEM NUMBER IN R0
BNE 1$ ;BRANCH IF IT IS NON-ZERO
MOV $ERRPC,-(KSP) ;PUT ERROR PC ON STACK FOR TYPING
TYPOC ;TYPE FAILING PC
BR 13$ ;GO TO RETURN
1$: CMPB #177,R0 ;SEE IF THIS IS THE PWR MON BIT ERROR ;DPM001
BNE 200$ ;BRANCH IF NOT TO CALL ERROR
MOV #PMBECW,R0 ;MOVE ADDRESS OF SPECIAL DATA HEADER TO R0
BR 210$ ;BRANCH TO CALL ERROR
200$: DEC R0 ;ADJUST ITEM NUMBER TO BE A POINTER
ASH #3,R0 ;LEFT SHIFT ITEM NO. 3 PLACES
BPL 22$ ;BRANCH IF ITEM NUMBER IS LESS THAN 200
CMP ERRCNT,#20 ;* SEE IF 20 (OCTAL) ERRORS HAVE PRINTED

```

```

1792 002102 002410          BLT      40$      ;:* BRANCH TO PRINT THE ERROR IF LESS
1793 002104 001404          BEQ      41$      ;:* BRANCH TO TYPE NO MORE DATA LINES IF EQUAL
1794 002106 062766 000004 000002  ADD     #4,2(KSP) ;:* CORRECT PC RETURN TO RETURN AFTER <CRLF> PRINT
1795 002114 000542          BR       13$      ;:* GO TO RETURN
1796 002116 104401 006421          TYPE     ,NOMORE ;:* TYPE MESSAGE TO ANNOUNCE NO MORE PRINTING OF ERRORS
1797 002122 000537          BR       13$      ;:* GO TO RETURN
1798 002124 022737 000001 001320 40$:  CMP     #1,ERRCNT ;:* SEE IF THIS IS THE FIRST ERROR
1799 002132 001415          BEQ     21$      ;:* BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
1800 002134 032777 000200 176774  BIT     #SW7,@SWR ;SEE IF SWITCH 7 IS UP
1801 002142 001404          BEQ     20$      ;BRANCH IF SWITCH NOT UP AND TYPE DATA
1802 002144 062766 000004 000002  ADD     #4,2(KSP) ;SKIP 'TYPE ,SCRLF' IF SW 7 IS UP
1803                                ;INHIBIT MULTIPLE ERROR TYPEOUTS
1804 002152 000523          BR       13$      ;BRANCH TO EXIT
1805 002154 042700 177400 20$:  BIC     #177400,RO ;CLEAR UPPER BYTE OF RO
1806 002160 062700 001672  ADD     #ER200+4,RO ;POINT TO DATA TABLE ENTRY
1807 002164 000426          BR       5$       ;GO TYPE DATA TABLE
1808 002166 042700 177000 21$:  BIC     #177000,RO ;CLEAR UPPER BYTE OF RO
1809 002172 062700 000300  ADD     #<ER200-$ERRTB>,RO ;ADD DIFFERENCE BETWEEN
1810                                ;ITEM 1 AND ITEM 201
1811                                ;:GET POINTER TO ERROR MESSAGE AND TYPE IT
1812                                ;:IF THE POINTER IS NOT ZERO
1813 002176 104401 001223 22$:  TYPE     ,SCRLF ;TYPE A <CRLF>
1814 002202 062700 001366  ADD     #$ERRTB,RO ;ADD BASE OF ERROR TABLE
1815 002206 012037 002216 210$: MOV     (RO)+,2$ ;P M MESSAGE POINTER IN TYPE STATEMENT
1816 002212 001404          BEQ     3$       ;BRANCH IF NO ERROR MESSAGE
1817 002214 104401          TYPE     ;TYPE ERROR MESSAGE
1818 002216 000000 2$:  .WORD   0 ;POINTER TO ERROR MESSAGE
1819 002220 104401 001223  TYPE     ,SCRLF ;TYPE CRLF
1820                                ;:GET THE POINTER TO THE DATA HEADER AND
1821                                ;:TYPE IT IF THE POINTER IS NOT ZERO
1822 002224 012037 002234 3$:  MOV     (RO)+,4$ ;PUT HEADER POINTER IN TYPE STATEMENT
1823 002230 001404          BEQ     5$       ;BRANCH IF NO DATA HEADER
1824 002232 104401          TYPE     ;TYPE THE DATA HEADER
1825 002234 000000 4$:  .WORD   0 ;POINTER TO DATA HEADER
1826 002236 104401 001223  TYPE     ,SCRLF ;TYPE CRLF
1827                                ;:THIS IS THE START OF THE DATA OUTPUT IF THE
1828                                ;:DATA POINTER IS NOT ZERO. RO POINTS TO THE
1829                                ;:DATA FORMAT, R1 POINTS TO THE ADDRESS OF
1830                                ;:THE DATA WORDS.
1831 002242 010146 5$:  MOV     R1,-(KSP) ;SAVE R1 ON THE STACK
1832                                TYPDAT=.
1833 002244 012001          MOV     (RO)+,R1 ;PUT DATA TABLE POINTER IN R1
1834 002246 001464          BEQ     12$      ;BRANCH IF NO DATA TABLE
1835 002250 012000          MOV     (RO)+,RO ;PICK UP DATA FORMAT POINTER
1836 002252 105710 6$:  TSTB   (RO) ;IS THIS WORD OCTAL
1837 002254 001003          BNE     7$       ;BRANCH IF NOT 16-BIT OCTAL
1838                                ;:WORD IS 16 BIT OCTAL FORMAT (DF = 0)
1839 00225 013146          MOV     @ (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1840 002260 104402          TYPOC ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
1841 002262 000451          BR       11$     ;GET READY FOR NEXT WORD
1842 002264 122710 000001 7$:  CMPB   #1,(RO) ;IS THE WORD DECIMAL
1843 002270 001003          BNE     8$       ;BRANCH IF NOT DECIMAL
1844                                ;:WORD IS DECIMAL FORMAT (DF = 1)
1845 002272 013146          MOV     @ (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1846 002274 104405          TYPDS ;TYPE THE WORD ON STACK AS DECIMAL
1847 002276 000443          BR       11$     ;GET READY FOR NEXT WORD
1848 002300 122710 000002 8$:  CMPB   #2,(RO) ;IS WORD 22-BIT PHYSICAL ADDRESS

```

```

1849 002304 001012          BNE      9$          ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
1850                          ;:WORD IS 22-BIT PHYSICAL FORMAT (DF = 2)
1851 002306 012146          MOV      (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1852 002310 004737 024040    JSR      PC,$DB20     ;CONVERT NUMBER TO OCTAL ASCIZ
1853 002314 062716 000003    ADD      #3,(KSP)     ;ONLY WANT 8 DIGITS
1854 002320 012637 002326    MOV      (KSP)+,30$   ;PUT POINTER AFTER 'TYPE' CALL
1855 002324 104401          TYPE     ;TYPE ASCIZ STRING
1856 002326 000000          .WORD   0            ;WORD HOLDS POINTER TO ASCIZ STRING
1857 002330 000426          BR       11$         ;GET READY FOR NEXT WORD
1858 002332 122710 000003    9$:     CMPB     #3,(R0) ;IS THIS A 16-BIT VIRTUAL ADDRESS
1859 002336 001004          BNE     10$         ;BRANCH IF NOT 16-BIT VIRT. ADDR.
1860                          ;:WORD IS 22-BIT VIRTUAL ADDRESS FORMAT
1861                          ;:KERNEL I-SPACE ASSUMED. (DF = 3)
1862 002340 013146          MOV      @ (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1863 002342 004737 002542    JSR      PC,TYPVAD    ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
1864 002346 000417          BR       11$         ;GET READY FOR NEXT WORD
1865 002350 122710 000004    10$:    CMPB     #4,(R0)   ;IS THIS A 16 BIT NUMBER TO BE CONVERTED TO
1866                          ;AN 18 BIT UNIBUS ADDRESS LEFT SHIFTED 6?
1867 002354 001003          BNE     100$        ;SKIP OVER FORMAT 4 ROUTINE IF NOT
1868                          ;:WORD IS FORMAT 4. DATA WORD IS A UNIBUS
1869                          ;:ADDRESS OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
1870
1871 002356 004737 002650    JSR      PC,UBADDR    ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
1872 002362 000411          BR       11$         ;GET READY FOR NEXT WORD
1873 002364 122710 000005    100$:   CMPB     #5,(R0)   ;IS THIS A 16 BIT NUMBER TO BE PRINTED AS
1874                          ;OCTAL WITH LEADING ZEROS SUPPRESSED?
1875 002370 001004          BNE     110$        ;BRANCH TO DECIMAL LEADING SPACES SUPPRESS ROUTINE
1876                          ;:WORD IS FORMAT 5. DATA WORD IS TO BE
1877                          ;:PRINTED IN OCTAL, LEADING ZEROS SUPPRESSED.
1878 002372 013146          MOV      @ (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1879 002374 104403          TYPOS   ;GO TYPE OCTAL SUPPRESS LEADING ZEROS
1880 002376 006           .BYTE   6            ;TYPE 6 DIGITS AND
1881 002377 000           .BYTE   0            ;SUPPRESS LEADING ZEROS
1882 002400 000402          BR       11$         ;GET READY FOR NEXT WORD
1883 002402          110$:           ;:WORD IS FORMAT 6. DATA WORD IS TO BE
1884                          ;:PRINTED IN DECIMAL, LEADING SPACES SUPPRESSED.
1885 002402 013146          MOV      @ (R1)+,-(KSP) ;PUSH NEXT 16-BIT WORD ON STACK
1886 002404 104405          TYPDS   ;POINT TO NEXT FORMAT BYTE
1887 002406 005200          11$:     INC      R0          ;TYPE TWO SPACES
1888 002410 104401 002426    TYPE    ,32$        ;IS THERE ANOTHER WORD?
1889 002414 005711          TST     (R1)        ;BRANCH IF NOT ALL DONE
1890 002416 001315          BNE     6$          ;RESTORE R1
1891 002420 012601          12$:    MOV      (KSP)+,R1 ;RESTORE R0
1892 002422 012600          13$:    MOV      (KSP)+,R0 ;RETURN TO ERROR ROUTINE
1893 002424 000207          RTS     PC          ;TWO SPACES
1894 002426 040 040 000 32$: .ASCIZ ? ?
1895                          .EVEN
1896 002432 002442 002476 002526 PMBECW: .WORD PMBECM,PMBECH,PMBECD,PMBECF ;4 WORDS POINTING TO BELOW
1897 002442 120 117 127 PMBECM: .ASCIZ ?POWER MONITOR BIT FOUND SET?
1898 002476 124 105 123 PMBECH: .ASCIZ ?TESTNO ERR PC CPUERR?
1899                          .EVEN
1900 002526 020146 001114 020700 PMBECB: .WORD $TESTN,$ERRPC,CPSAVE,0
1901 002536 000 000 000 PMBECF: .BYTE 0,0,0,0

```

1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914 002542 104411
 1915 002544 016601 000002
 1916 002550 005000
 1917 002552 073027 000003
 1918 002556 006300
 1919 002560 006001
 1920 002562 006001
 1921 002564 006001
 1922 002566 062700 172340
 1923 002572 011003
 1924 002574 005002
 1925 002576 073227 000006
 1926 002602 060103
 1927 002604 005502
 1928 002606 010237 001230
 1929 002612 010337 001226
 1930 002616 012746 001226
 1931 002622 004737 024040
 1932 002626 062716 000003
 1933 002632 012637 002640
 1934 002636 104401
 1935 002640 000000
 1936
 1937 002642 104412
 1938 002644 012616
 1939 002646 000207

.SBTTL CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS

THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED TO ASCIZ AND TYPED.

```
TYPVAD: SAVREG      ;SAVE ALL REGISTERS
          MOV        2(KSP),R1    ;PUT VIRTUAL ADDR IN R1
          CLR        R0           ;CLEAR R0 FOR CALCULATIONS
          ASHC      #3,R0        ;LEFT SHIFT R0,R1 3 PLACES
          ASL        R0           ;LEFT SHIFT R0 ONE MORE PLACE
          ROR        R1           ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
          ROR        R1           ;RIGHT SHIFT R1
          ROR        R1           ;RIGHT SHIFT R1
          ADD        #KIPAR0,R0   ;FORM DESIRED PAR ADDR IN R0
          MOV        (R0),R3     ;PUT CONTENTS OF PAR IN R3
          CLR        R2           ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
          ASHC      #6,R2        ;LEFT SHIFT <R2,R3> 6 PLACES
          ADD        R1,R3        ;ADD OFFSET IN R1 TO BASE IN R3
          ADC        R2           ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
          MOV        R2,PADRSH   ;PUT UPPER 6 BITS OF ADDR IN CORE
          MOV        R3,PADRSL   ;PUT LOWER 16 BITS OF ADDR IN CORE
          MOV        #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK
          JSR        PC,$DB20    ;CONVERT NUMBER TO OCTAL ASCIZ
          ADD        #3,(KSP)    ;ONLY TYPE 8 DIGITS
          MOV        (KSP)+,3$   ;PUT POINTER AFTER TYPE INST
          TYPE      ;TYPE THE 22-BIT VIRTUAL ADDRESS
3$:      .WORD      0           ;THIS WORD HOLDS THE POINTER TO
          ;THE ASCIZ STRING
          RESREG    ;RESTORE ALL THE REGISTERS
          MOV        (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDR ON STACK
          RTS       PC          ;RETURN TO ERROR HANDLER
```

```

1940
1941
1942
1943
1944
1945
1946
1947 002650 104411
1948 002652 016601 000002
1949 002656 005000
1950 002660 073027 000006
1951 002664 010137 001226
1952 002670 010037 001230
1953 002674 012746 001226
1954 002700 004737 024040
1955 002704 062716 000005
1956 002710 012637 002716
1957 002714 104401
1958 002716 000000
1959 002720 104412
1960 002722 012616
1961 002724 000207
  
```

```

.SBTTL SUBROUTINE TO CONVERT WORD TO A UNIBUS ADDRESS AND TYPE
:*****
:*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
:*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
:*6 DIGIT NUMBER. IT USES R1 & R0 AND LEAVES
:*ALL OTHER REGISTERS UNCHANGED.
:*****
UBADDR: SAVREG
      MOV      2(KSP),R1      ;LOAD 16 BIT ADDRESS INTO R1
      CLR      R0            ;CLEAR R0 FOR CALCULATIONS
      ASHC     #6,R0         ;LEFT SHIFT <R0:R1> 6 PLACES
      MOV      R1,PADRSL     ;PUT LOWER 16 BITS IN PADRSL
      MOV      R0,PADRSH     ;PUT UPPER 6 BITS IN PADRSH
      MOV      #PADRSL,-(KSP) ;PUSH POINTER TO WORDS ON STACK
      JSR      PC,$DB20      ;JUMP TO CONVERT ROUTINE
      ADD      #5,(KSP)       ;ONLY USE LOWER 6 CHARS.
      MOV      (KSP)+,3$     ;PUT POINTER AFTER TYPE CALL.
      TYPE
3$:   .WORD    0              ;HOLDS POINTER TO FIRST CHAR.
      RESREG
      MOV      (KSP)+,(KSP)  ;LEAVE ONLY RETURN ADDRESS ON STACK
      RTS      PC           ;RETURN TO ERROR TYPE ROUTINE.
  
```

1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978 002726
1979 002734
1980 002736
1981 002744
1982 002752

000020
032766 000020 000002
001406
016637 000002 001346
042766 000020 000002
000006

```
.SBTTL  TURN OFF AND SAVE T-BIT
:*****
:*****
:*          **SUBROUTINES UNIQUE TO THIS PROGRAM**
:*****
:*****
:*
:*          THIS TRAP ROUTINE IS REACHED BY THE TRAP CALL 'TBITO'.  IT IS
:*          USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
:*          IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
:*          PREVIOUS STATUS WHEN CONDITIONS WARRANT.
:*****
:*****
TBIT=BIT4          ;T-BIT IS BIT04 IN PROC. STATUS
TBITOF: BIT      #TBIT,2(KSP) ;IS THE T-BIT ON?
          BEQ     1$          ;BRANCH TO EXIT IF IT IS NOT ON
          MOV     2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING T BIT
          BIC     #TBIT,2(KSP) ;CLEAR T BIT
1$:          RTT             ;RETURN TO PROGRAM
```

```
1983 .SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION
1984 :*****
1985 :*
1986 :* THIS TRAP ROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR'. IT IS
1987 :* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
1988 :* BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR STATUS STORED
1989 :* IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
1990 :* AND DOES AN 'RTT'.
1991 :*
1992 :*****
1993 002754 013766 001346 000002 TBITRE: MOV OLDPSW,2(KSP) ;PUT OLD PSW ON STACK
1994 002762 042737 000020 001346 BIC #TBIT,OLDPSW ;CLEAR T-BIT IN 'OLDPSW'
1995 ;SO THAT IT WON'T BE TURNED ON BY ACCIDENT
1996 002770 000006 RTT ;RETURN TO PROGRAM AND INHIBIT T-BIT TRAP AFTER THIS INSTRUCTION
```


1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008 002772 012703 170200
2009 002776 005023
2010 003000 005023
2011 003002 032737 000040 172516
2012 003010 001402
2013 003012 012723 020000
2014 003016 005023
2015 003020 022703 170400
2016 003024 001374
2017 003026 000207

```
.SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
:*****
:
:   THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS IF MAPPING IS
:   DISABLED BY LOADING THE ADDRESS OF MAPLO0 INTO R3 AND THEN
:   CLEARING THE REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE
:   MAPH37. IF MAPPING IS ENABLED, ALL REGISTERS EXCEPT MAPL1
:   IS CLEARED. THE LOWER WORD OF MAPL1 RECEIVES 20000. THIS IS
:   SO APT CAN PROPERLY MONITOR THE PROGRESS OF THE DIAGNOSTIC.
:*****
CLRMAP: MOV    #MAPLO,R3      ;PUT FIRST MAP ADDR IN R3
        CLR    (R3)+        ;CLEAR MAPLO
        CLR    (R3)+        ;CLEAR MAPLO+2
        BIT    #BITS,MMR3   ;SEE IF MAPPING IS ENABLED
        BEQ    1$           ;BRANCH TO CLEAR ALL IF NOT ENABLED
        MOV    #20000,(R3)+ ;LOAD 20000 INTO MAPL1 FOR POSSIBLE APT USE
1$:     CLR    (R3)+        ;CLEAR MAP REGISTERS
        CMP    #MAPH37+2,R3 ;SEE IF LAST ADDR+2 IS IN R3
        BNE    1$          ;BRANCH IF NOT DONE YET
        RTS    PC          ;RETURN TO MAIN PROGRAM
```

```

2018 .SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
2019 :*****
2020 :*
2021 :* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
2022 :* REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS. IT
2023 :* KEEPS A 'LOGICAL AND' AND A 'LOGICAL OR' OF EACH ADDRESS THAT
2024 :* TIMES OUT.
2025 :*
2026 :*****
2027 003030 005227 TIMEOUT:INC (PC)+ ;INCREMENT ONE TIME GATE
2028 003032 177777 TOFLAG: .WORD -1 ;ONE TIME ENTANCE FLAG
2029 003034 001403 BEQ 10$ ;BRANCH IF FLAG IS NOW ZERO
2030 003036 005237 020142 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
2031 003042 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE FIRST ERROR.
2032 ;THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE
2033 ;FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
2034 003044 012637 001342 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
2035 003050 012637 001344 MOV (KSP)+,OLDPS ;SAVE OLD PSW
2036 003054 105737 007140 TSTB CPUTYP ;SEE IF THIS IS AN 11/44
2037 003060 001406 BEQ 1$ ;BRANCH TO CONTINUE IF IT IS
2038 003062 005237 001330 INC PCPUER ;INCREMENT PCPUER TO SHOW A TIMEOUT OCCURED
2039 003066 005737 001326 TST CPUEXP ;SEE IF THERE WAS AN EXPECTED ERROR
2040 003072 001435 BEQ 3$ ;GO REPORT ERROR IF NONE EXPECTED
2041 003074 000442 BR 4$ ;BRANCH TO EXIT IF TIMEOUT WAS EXPECTED
2042 003076 013737 177766 001330 1$: MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
2043 003104 013737 001330 177766 MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
2044 003112 023737 001330 001326 CMP PCPUER,CPUEXP ;SEE IF EXPECTED CONDITION CAME UP.
2045 003120 001405 BEQ 2$ ;BRANCH IF IT WAS A TIMEOUT
2046 003122 012737 177777 003032 MOV #-1,TOFLAG ;RESET ONE TIME GATE
2047 003130 104001 ERROR +1 ;NOT THE CORRECT CPU TRAP THROUGH 4
2048 003132 000423 BR 4$ ;BRANCH TO EXIT
2049 003134 105737 007140 2$: TSTB CPUTYP ;IS THIS AN 11/24?
2050 003140 001403 BEQ 25$ ;BRANCH IF NOT
2051 003142 005237 001320 INC ERRCNT ;COUNT THIS AS A TIMEOUT
2052 003146 000415 BR 4$ ;GO TO EXIT
2053 003150 022737 000020 001326 25$: CMP #TIMOUT,CPUEXP ;SEE IF A TIMEOUT WAS EXPECTED
2054 003156 001411 BEQ 4$ ;BRANCH TO EXIT THIS ROUTINE IF IT WAS
2055 003160 010046 MOV RO,-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
2056 003162 013746 172356 MOV KIPAR7,-(SP) ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
2057 003166 004737 003674 3$: JSR PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2058 003172 012737 177777 003032 MOV #-1,TOFLAG ;RESET ONE TIME GATE
2059 003200 104201 ERROR +201 ;THE FOLLOWING REGISTERS TIMED OUT WHEN READ
2060 003202 012737 177777 003032 4$: MOV #-1,TOFLAG ;RESET ONE TIME GATE
2061 003210 013746 001344 MOV OLDPS,-(KSP) ;RESTORE OLD PSW
2062 003214 013746 001342 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON THE STACK
2063 003220 000006 RTT ;RETURN TO THE TEST

```

```

2064 .SBTTL CPU TRAP HANDLER ROUTINES
2065 :*****
2066 :*                                     **TRAP HANDLING ROUTINES**
2067 :*****
2068
2069 :*****
2070 :*
2071 :* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THROUGH
2072 :* 'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
2073 :* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
2074 :* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
2075 :* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
2076 :* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
2077 :* 'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
2078 :* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
2079 :* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
2080 :*
2081 :*****
2082 003222 005227 CPUER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
2083 003224 177777 CPFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG
2084 003226 001403 BEQ 10$ ;BRANCH IF FIRST TIME IN
2085 003230 005237 020142 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
2086 003234 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
2087 :I FINISHED REPORTING THE FIRST ERROR. THE SECOND ENTRY ADDRESS IS ON
2088 :THE STACK, AND THE FIRST ERROR CONDITION IS PROBABLY STILL LOCKED UP.
2089 003236 012637 001342 10$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
2090 003242 012637 001344 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
2091 003246 013737 177766 001330 MOV CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
2092 003254 013737 001342 001340 MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT
2093 003262 005737 001326 TST CPUEXP ;SEE IF ANY CONDITION WAS EXPECTED
2094 003266 001414 BEQ 1$ ;BRANCH IF NO TRAP WAS EXPECTED
2095 003270 105737 007140 TSTB CPUTYP ;SEE IF THIS WAS AN 11/44
2096 003274 001016 BNE 2$ ;BRANCH TO CONTINUE IF AN 11/24
2097 003276 023737 001330 001326 CMP PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
2098 003304 001417 BEQ 3$ ;BRANCH IF ERROR CODES MATCH
2099 003306 012737 177777 003224 MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2100 003314 104001 ERROR +1 ;NOT THE CORRECT CPU TRAP THROUGH 4
2101 003316 000412 BR 3$ ;SKIP NEXT INSTRUCTION
2102 003320 012737 177777 003224 1$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2103 003326 104002 ERROR +2 ;UNEXPECTED CPU TRAP THROUGH 4
2104 003330 000405 BR 3$ ;SKIP NEXT INSTRUCTION
2105 003332 005237 001320 2$: INC ERRCNT ;INCREMENT ERRCNT TO SHOW AN ERROR FOR 11/24
2106 003336 013737 001326 001330 MOV CPUEXP,PCPUER ;PUT EXPECTED CONTENTS IN PCPUER
2107 003344 012737 177777 003224 3$: MOV #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
2108 003352 013737 001330 177766 MOV PCPUER,CPUERR ;CLEAR CPU ERROR REGISTER
2109 003360 013746 001344 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
2110 003364 013746 001342 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
2111 003370 000006 RTT ;RETURN FROM INTERRUPT OR ABORT

```

```

2112 .SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
2113 :*****
2114 :*
2115 :* THIS ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAPS
2116 :* AND ABORTS. IT WILL REPORT THE CONDITION OF ALL THE MEMORY
2117 :* MANAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
2118 :* TRY TO CONTINUE RUNNING.
2119 :*
2120 :*****
2121 003372 005227 MMTRAP: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
2122 003374 177777 MMFLAG: .WORD -1 ;FLAG SHOULD BE NEG ONE
2123 003376 001403 BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE
2124 003400 005237 020142 INC $MSGTY ;INDICATE TO APT A FATAL ERROR OCCURED
2125 003404 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE I FINISHED REPORTING THE
2126 :FIRST ERROR. THE SECOND ENTRY ADDRESS IS ON THE STACK AND THE FIRST ERROR
2127 :CONDITION IS PROBABLY STILL LOCKED UP .
2128 003406 011637 001340 10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP
2129 003412 012637 001342 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
2130 003416 012637 001344 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
2131 003422 013737 177572 001350 MOV MMR0,PMMR0 ;SAVE STATUS REGISTER
2132 003430 013737 177574 001352 MOV MMR1,PMMR1 ;SAVE AUTO INC/DEC REGISTER
2133 003436 013737 177576 001354 MOV MMR2,PMMR2 ;SAVE VIRTUAL ADDRESS REGISTER
2134 003444 104003 ERROR +3 ;UNEXPECTED M.M. ABORT OR TRAP
2135 003446 042737 177776 177572 1$: BIC #177776,MMR0 ;CLEAR ALL BITS EXCEPT 0
2136 003454 012737 177777 003374 MOV #-1,MMFLAG ;RESTORE A NEGATIVE ONE TO FLAG
2137 003462 013746 001344 MOV OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
2138 003466 013746 001342 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK
2139 003472 000006 RTT ;RETURN TO MAIN PROGRAM
    
```

```

2140 .SBTTL SUBROUTINE TO TEST A LOCATION FOR WRITEABILITY
2141 :*****
2142 :*
2143 :* THIS SUBROUTINE CLEARS A TEST LOCATION, LOADS THE LOCATION USING
2144 :* THE MAP REGISTER, AND DETERMINES IF THE LOCATION WAS LOADED. IF
2145 :* IT WAS, RETURN IS NORMAL TO THE TEST. IF NOT, THE PC ON THE
2146 :* STACK IS UPDATED BY 2 AND THEN A RETURN IS EXECUTED.
2147 :*
2148 :*****
2149 003474 005237 001174 TSTLOC: INC $TMP0 ;INCREMENT REGISTER COUNTER
2150 003500 005737 005774 TST FLOATR ;SEE IF BIT 15 OF FLOATR IS SET
2151 003504 100011 BPL 1$ ;BRANCH IF STILL PLUS
2152 003506 022705 020230 CMP #DDW1,R5 ;SEE IF R5 IS POINTING TO UPPER DDW
2153 003512 001417 BEQ NEXT ;BRANCH IF SO - ALL DONE
2154 003514 012705 020230 MOV #DDW1,R5 ;MOVE ADDRESS OF DDW1 TO R5 AND
2155 003520 012737 000001 005774 MOV #BIT0,FLOATR ;RESET BIT 0 IN FLOATR
2156 003526 000411 BR NEXT ;BRANCH OVER ASL
2157 003530 005227 1$: INC (PC)+ ;INCREMENT NEXT LOCATION FOR FIRST TIME THROUGH CHECK
2158 003532 177777 FTTHRU: .WORD -1 ;FIRST TIME ENTRANCE FLAG
2159 003534 001004 BNE 1$ ;BRANCH IF NOT FIRST TIME
2160 003536 012737 000001 005774 MOV #BIT0,FLOATR ;MOVE BIT 0 TO LOCATION FLOATR
2161 003544 000402 BR NEXT ;BRANCH OVER THE ASL
2162 003546 006337 005774 1$: ASL FLOATR ;ROTATE THE TEST BIT TO THE LEFT
2163 003552 005037 037776 NEXT: CLR 37776 ;CLEAR TEST LOCATION
2164 003556 005037 001330 CLR PCPUER ;CLEAR ERROR LOCATION
2165 003562 010210 MOV R2,(R0) ;TRY TO LOAD TEST CELL THROUGH MAP
2166 003564 023702 037776 CMP 37776,R2 ;SEE IF TEST LOCATION WAS LOADED
2167 003570 001414 BEQ 2$ ;BRANCH IF IT WAS LOADED
2168 003572 004737 005076 JSR PC,CHKLMA ;GO SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
2169 003576 177736 177734 .WORD LMAHI,LMALOW ;ADDRESSES OF LMA REGISTERS
2170 003602 000407 BR 2$ ;RETURN IS HERE IF OK
2171 003604 005737 001330 TST PCPUER ;SEE IF A TIMEOUT OCCURED
2172 003610 001402 BEQ 1$ ;BRANCH OVER SPECIAL STACK PUSH IF NOT
2173 003612 013743 001174 MOV $TMP0,-(R3) ;PUSH REGISTER NUMBER THAT TIMED OUT ON SPECIAL STACK
2174 003616 062716 000002 1$: ADD #2,(SP) ;CORRECT PC RETURN FOR LOAD FAILURE INDICATION
2175 003622 000207 2$: RTS PC ;RETURN FROM THIS SUBROUTINE

```

```
2176 .SBTTL SUBROUTINE TO LOAD DATAOR AND DATAND
2177 :*****
2178 :*
2179 :* THIS SUBROUTINE ASSUMES THE DATA TO BE ANDED AND ORED HAS BEEN PUT
2180 :* ON THE STACK BEFORE THIS SUBROUTINE WAS CALLED. IT BIT SETS THE
2181 :* DATA ONTO DATAOR, COMPLEMENTS THE DATA AND BIT CLEARS IT ONTO
2182 :* DATAND.
2183 :*
2184 :*****
2185 003624 056637 000002 001246 DATEXT: BIS 2(SP),DATAOR ;SET THE 'OR' PATTERN TO DATAOR
2186 003632 005166 000002 COM 2(SP) ;COMPLIMENT THE DATA
2187 003636 046637 000002 001242 BIC 2(SP),DATAND ;CLEAR THE 'AND' PATTERN TO DATAND
2188 003644 012616 MOV (SP)+,(SP) ;CLEAN THE STACK FOR THE RETURN
2189 003646 000207 RTS PC ;RETURN
```

```
2190 .SBTTL SUBROUTINE TO LOAD PATAOR AND PATAND
2191 :*****
2192 :*
2193 :* THIS SUBROUTINE ASSUMES THE DATA TO BE ANDED AND ORED HAS BEEN PUT
2194 :* ON THE STACK BEFORE THIS SUBROUTINE WAS CALLED. IT BIT SETS THE
2195 :* DATA ONTO PATTOR, COMPLEMENTS THE DATA AND BIT CLEARS IT ONTO
2196 :* PATAND.
2197 :*
2198 :*****
2199 003650 056637 000002 001254 PATEXT: BIS 2(SP),PATTOR ;SET THE 'OR' PATTERN TO PATTOR
2200 003656 005166 000002 COM 2(SP) ;COMPLIMENT THE PATTERN
2201 003662 046637 000002 001252 BIC 2(SP),PATAND ;CLEAR THE 'AND' PATTERN TO PATAND
2202 003670 012616 MOV (SP)+,(SP) ;CLEAN UP STACK FOR RETURN
2203 003672 000207 RTS PC ;RETURN
```

```

2204 .SBTTL SUBROUTINE TO TAKE PAR AND LOAD 2 WORDS EACH OF ADDROR & ADRAND
2205 :*****
2206 :*
2207 :* THIS SUBROUTINE ASSUMES THE CONTENTS OF THE PAR, AND THE VIRTUAL
2208 :* ADDRESS HAVE BEEN PUT ON THE STACK. IT TAKES THE PAR, SHIFTS IT
2209 :* TO EXPOSE THE UPPER ADDRESS BITS, BIT SETS THEM TO ADDROR+2, COM-
2210 :* PLIMENTS THE CONTENTS AND BIT CLEARS ADRAND+2. AFTER RELOADING
2211 :* THE PAR IN R5, IT SHIFTS TO GET THE LOWER 16 BIT EQUIVALENT AND
2212 :* ADDS THE VIRTUAL ADDRESS TO CREATE THE PHYSICAL LOWER 16 BITS.
2213 :* THEN IT BIT SETS THEM TO ADDROR, COMPLIMENTS THE CONTENTS AND
2214 :* BIT CLEARS ADRAND. ANOTHER COMPLIMENT BRINGS THE STATE BACK TO
2215 :* ITS ORIGINAL STATE. THIS SUBROUTINE LEAVES WITH THE LOWER 16
2216 :* BITS AND THE UPPER 6 BITS ON THE STACK, AND ARE TO BE REMOVED IN
2217 :* THAT ORDER, AND MUST BE REMOVED AFTER RETURN.
2218 :*
2219 :*****
2220 ADREXT: MOV R5, -(SP) ;SAVE R5
2221 MOV 4(SP), R5 ;MOVE PAR CONTENTS TO R5 FOR SHIFTING
2222 ASH #-10, R5 ;SHIFT R5 TO THE RIGHT 10 PLACES
2223 BIC #177700, R5 ;CLEAR BITS 15 TO 6
2224 CMP #74, R5 ;SEE IF I/O PAGE
2225 BNE 1$ ;BRANCH IF NOT
2226 MOV #77, R5 ;RESET R5 TO 77
2227 1$: MOV R5, $TMP5 ;MOVE OBTAINED UPPER 6 BITS TO $TMP5 FOR FUTURE TRANSFER
2228 BIS R5, ADDROR+2 ;SET THE 'OR' PATTERN OF UPPER 6 BITS TO ADDROR+2
2229 COM R5 ;COMPLIMENT R5
2230 BIC R5, ADRAND+2 ;CLEAR THE 'AND' PATTERN OF UPPER 6 BITS TO ADRAND+2
2231 MOV 4(SP), R5 ;PUT PAR CONTENTS BACK IN R5
2232 MOV $TMP5, 4(SP) ;MOVE UPPER 6 BITS OF PHYSICAL ADDRESS ON STACK
2233 CMP #74, $TMP5 ;SEE IF I/O PAGE
2234 BNE 2$ ;BRANCH IF NOT
2235 MOV #77, EADRES+2 ;SET 77 IN UPPER ERROR LOCATION
2236 MOV 6(SP), EADRES ;SET LOWER ADDRESS IN LOWER ERROR LOCATION
2237 BR 3$ ;BRANCH OVER PREP
2238 2$: BIC #176000, R5 ;STRIP OFF UPPER 6 BITS OF PAR CONTENTS
2239 ASH #6, R5 ;SHIFT REMAINING BITS 6 PLACES TO THE LEFT
2240 BIC #160000, 6(SP) ;STRIP OFF PAR PAGE BITS FROM ADDRESS TO FORM OFFSET
2241 ADD R5, 6(SP) ;FORM LOWER 16 BITS OF ADDRESS
2242 3$: MOV (SP)+, R5 ;RESTORE R5
2243 BIS 4(SP), ADDROR ;SET THE 'OR' PATTERN TO ADDROR
2244 COM 4(SP) ;COMPLIMENT THE ADDRESS
2245 BIC 4(SP), ADRAND ;CLEAR THE 'AND' PATTERN TO ADRAND
2246 COM 4(SP) ;RETURN ADDRESS TO ITS ORIGINAL STATE
2247 CMP #77, EADRES+2 ;SEE IF I/O PAGE IN UPPER LOCATION
2248 BEQ 4$ ;BRANCH TO EXIT IF SO
2249 MOV 4(SP), EADRES ;PUT LOWER 16 BITS IN ERROR STATUS LOCATION
2250 MOV 2(SP), EADRES+2 ;PUT UPPER 6 BITS IN ERROR STATUS LOCATION
2251 4$: MOV (SP)+, 2(SP) ;PUT RETURN ADDRESS WHERE IT BELONGS
2252 TST (SP)+ ;CLEAN STACK
2253 RTS PC ;RETURN
  
```



```

2254 .SBTTL SUBROUTINE TESTING RELOCATION ADDER
2255 :*****
2256 :* THE FOLLOWING SUBROUTINE IS USED IN TESTS 13 AND 16, AND USES THE DATA
2257 :* BANK FOLLOWING TO EXECUTE THE LOOPS IN THE TEST. R3 IS THE SPECIAL
2258 :* 'STACK' POINTER, INITIALIZED AT THE BEGINING TO THE R3STAK DATA BANK.
2259 :* THE STACK POINTER IS ADVANCED 3 WORDS FOR EACH PASS. CALL THIS SUB-
2260 :* ROUTINE IN THIS MANNER:
2261 :* CLR $TMP4 ;CLEAR $TMP4 - USED IN ERROR RETURN
2262 :*1$: JSR PC,MAPADD ;GO DO THE TEST
2263 :* ERROR +ERRORNUMBER ;RETURN IS HERE FOR ERROR
2264 :* BR 1$ ;BRANCH BACK TO CONTINUE TEST
2265 :*****
2266 004106 005737 004312 MAPADD: TST LOEFLG ;SEE IF THIS ENTRY WAS AN ERROR
2267 004112 001070 BNE 3$ ;GO CONTINUE TEST
2268 004114 011646 MOV (SP),-(SP) ;MOVE RETURN UP ONE NOTCH
2269 004116 062766 000004 000002 ADD #4,2(SP) ;CREATE ADDRESS ON STACK FOR FINAL RETURN
2270 004124 012703 004314 MOV #R3STAK,R3 ;SET THE SPECIAL STACK POINTER
2271 004130 012704 000013 MOV #13,R4 ;SET THE LOOP COUNTER
2272 004134 012737 004162 001106 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2273 004142 005077 175134 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
2274 004146 012377 175126 1$: MOV (R3)+,@LREGL ;LOAD LOWER BITS OF MAPPING REG
2275 004152 013737 001256 172354 MOV LOWEST,KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
2276 004160 012300 MOV (R3)+,R0 ;SELECT PAR6, OFFSET IS AUGEND
2277 004162 011001 2$: MOV (R0),R1 ;READ LOCATION DEFINED BY 4TH WORD IN TABLE
2278 004164 012337 001310 MOV (R3)+,UBM24L ;MOVE ANTICIPATED PHYSICAL ADDRESS TO UBM24L
2279 004170 020137 001310 CMP R1,UBM24L ;SEE IF THE MAP'S FETCH WAS CORRECT
2280 004174 001443 BEQ 4$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
2281 004176 004737 005076 JSR PC,CHKLMA ;GO SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
2282 004202 177736 177734 .WORD LMAHI,LMALOW ;ADDRESSES OF LMA REGISTERS
2283 004206 000436 BR 4$ ;RETURN IS HERE IF OK
2284 004210 032777 004000 174720 BIT #BIT11,@SWR ;CHECK TO SEE IF 11/24 WITH UB MEMORY ONLY
2285 004216 001406 BEQ 25$ ;BRANCH IF NOT
2286 004220 017737 175064 001176 MOV @UBM24L,$TMP1 ;GET EXPECTED DATA FOR ERROR CALL
2287 004226 004737 005000 JSR PC,PTMP2 ;GO PREPARE $TMP2 FOR ERROR CALL
2288 004232 000404 BR 26$ ;GO CALL ERROR
2289 004234 011037 001176 25$: MOV (R0),$TMP1 ;GET EXPECTED DATA FOR ERROR CALL
2290 004240 010137 001200 MOV R1,$TMP2 ;GET DATA FROM R1 FOR ERROR CALL
2291 004244 162703 000002 26$: SUB #2,R3 ;ANTICIPATE ERROR LOOPING BY UNDOING AUTOINC
2292 004250 013737 001310 005764 MOV UBM24L,EADRES ;PUT ADDRESS IN EADRES FOR ERROR CALL
2293 004256 005037 005766 CLR EADRES+2 ;CLEAR UPPER LOCATION
2294 004262 011646 MOV (SP),-(SP) ;PUT AN EXTRA RETURN ON FOR POSSIBLE ERROR LOOPING
2295 004264 012737 000001 004312 MOV #1,LOEFLG ;SET FLAG SHOWING ERROR CALL WAS CALLED
2296 004272 000207 RTS PC ;EXIT TO ERROR CALL AT TEST
2297 004274 062703 000002 3$: ADD #2,R3 ;RESTORE AUTOINC, ERROR LOOPING NOT DONE
2298 004300 062706 000002 ADD #2,SP ;CLEAN EXTRA RETURN OFF STACK - LOOPING NOT DONE
2299 004304 077460 4$: SOB R4,1$ ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT 0
2300 004306 005726 TST (SP)+ ;EXPOSE NEXT TEST RETURN ADDRESS
2301 004310 000207 RTS PC ;EXIT TO NEXT TEST
2302 004312 000000 LOEFLG: .WORD 0 ;LOOP ON ERROR FLAG LOCATION

```

2303
 2304
 2305
 2306
 2307
 2308
 2309 004314 060000 140000 060000
 2310 004322 052524 145252 057776
 2311 004330 045252 152524 057776
 2312 004336 050420 150420 061040
 2313 004344 054630 144210 061040
 2314 004352 044210 154630 061040
 2315 004360 056734 142104 061040
 2316 004366 042104 156734 061040
 2317 004374 057776 141042 061040
 2318 004402 041042 157776 061040
 2319 004410 057776 140002 060000

:DATA IN THE R3STAK DATA BANK BELOW IS ARRANGED IN THE FOLLOWING ORDER:
 :>>>NOTE<<<: THE 'OFFSET' COLUMN IS NOT IN THE STACK DUE TO THE DELIMITER
 (:) BETWEEN THE EXPECTED ADDRESS AND THE OFFSET VALUES

	BASE	!VIRTUAL!	!PHYSCL!	!EXPCTD!	OFFSET	R4 VALUE
R3STAK: .WORD	060000	140000	060000	060000	000000	R4=13
.WORD	052524	145252	057776	005252		R4=12
.WORD	045252	152524	057776	012524		R4=11
.WORD	050420	150420	061040	010420		R4=10
.WORD	054630	144210	061040	004210		R4=7
.WORD	044210	154630	061040	014630		R4=6
.WORD	056734	142104	061040	002104		R4=5
.WORD	042104	156734	061040	016734		R4=4
.WORD	057776	141042	061040	001042		R4=3
.WORD	041042	157776	061040	017776		R4=2
.WORD	057776	140002	060000	000002		R4=1

```

2320          .SBTTL  SUBROUTINE TO TEST CARRY PROP OF MAP'S RELOC ADDER
2321          :*****
2322          :* THIS SUBROUTINE IS USED BY TESTS 14 AND 17.  CODE CALLING THIS SUB-
2323          :* ROUTINE IS AS FOLLOWS:
2324          :* CLR      $TMP4          ;CLEAR $TMP4 - USED IN ERROR RETURN
2325          :*LAB: JSR      PC,TCPMRA ;GO DO THE TEST
2326          :* ERROR  +211          ;RETURN IS HERE IF AN ERROR
2327          :* BR      LAB          ;BRANCH BACK TO CONTINUE TEST
2328          :*****
2329 004416 005737 004312 TCPMRA: TST    LOEFLG ;TEST ERROR FLAG TO SEE IF THIS ENTRY IS FROM ERROR
2330 004422 001132          BNE    4$ ;BRANCH TO CONTINUE TEST IF SO
2331 004424 011646          MOV    (SP),-(SP) ;MOVE RETURN ADDRESS UP ONE NOTCH
2332 004426 062766 000004 000002 ADD    #4,2(SP) ;CREATE CORRECT FINAL RETURN ADDRESS
2333 004434 012737 177777 004674 MOV    #-1,35$ ;INITIALIZE FLAG AS NEGATIVE ONE
2334 004442 005077 174634          CLR    @LREGU ;CLEAR UPPER 6 BITS OF MAP REG
2335 004446 012777 020000 174624 MOV    #20000,@LREGL ;LOAD 4K BASE INTO MAP REGISTER
2336 004454 012701 100100          MOV    #100100,R1 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
2337 004460 012700 150000          MOV    #150000,R0 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
2338 004464 012737 000277 172350 MOV    #277,KIPAR4 ;START WITH PHYSICAL 6K
2339 004472 013737 001256 172354 MOV    LOWEST,KIPAR6 ;LOAD PAR 6 WITH MAP REG'S ADDR
2340 004500 012737 004552 001106 MOV    #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
2341 004506 005037 001330          1$: CLR    PCPUER ;CLEAR TIME OUT FLAG
2342 004512 062737 010000 001310 ADD    #10000,UBM24L ;FORM EXPECTED LMA FOR POSSIBLE USE
2343 004520 001002          BNE    15$ ;BRANCH AROUND UBM24U INCREMENT IF NOT ZERO
2344 004522 005237 001312          INC    UBM24U ;INCREMENT UPPER LOCATION
2345 004526 012737 000020 001326 15$: MOV    #20,CPUEXP ;EXPECTING A UNIBUS TIME OUT DURING TEST
2346 004534 013710 001364          MOV    DATA,(R0) ;THIS LOAD WILL TIME OUT WHEN YOU HAVE REACHED THE TOP
2347          ;OF MEMORY IT SELECTS PAR 6 WHICH WILL PUT ADDR <XXX1>0000 ON THE UNIBUS. THE X'S WILL
2348          ;SELECT THE LOWEST USABLE MAPPING REGISTER. THE DEFAULT CASE IS 00010000, SELECTING
2349          ;MAP REGISTER 0.
2350 004540 005037 001326          CLR    CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2351 004544 005737 001330          TST    PCPUER ;SEE IF THERE WAS MAIN MEMORY
2352 004550 001050          BNE    3$ ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
2353 004552 012737 000040 001326 2$: MOV    #NEXMEM,CPUEXP ;POSSIBLE CACHE NON-EXISTENT MEMORY
2354 004560 011103          MOV    (R1),R3 ;READ TEST LOCATION VIA FASTBUS
2355 004562 005037 001326          CLR    CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2356 004566 022737 000040 001330 CMP    #NEXMEM,PCPUER ;WAS THIS CACHE NON-EXISTENT MEMORY
2357 004574 001436          BEQ    3$ ;BRANCH IF NON-EXISTENT MEMORY
2358 004576 011002          MOV    (R0),R2 ;READ TEST LOCATION VIA UNIBUS MAP
2359 004600 020203          CMP    R2,R3 ;COMPARE TEST DATA R2=MAP DATA, R3=FASTBUS DATA
2360 004602 001433          BEQ    3$ ;BRANCH IF IT WAS THE SAME
2361 004604 004737 005076          JSR    PC,CHKLMA ;GO CHECK FOR 11/24 WITH UB MEMORY
2362 004610 177736 177734          .WORD LMAHI,LMALOW ;ADDRESSES OF LMA REGISTERS
2363 004614 000426          BR    3$ ;RETURN IS HERE IF OK
2364 004616 032777 004000 174312 BIT    #BIT11,@SWR ;SEE IF THIS IS AN 11/24 WITH UB MEMORY ONLY
2365 004624 001403          BEQ    25$ ;BRANCH IF NOT
2366 004626 004737 005000          JSR    PC,PTMP2 ;GO PREPARE $TMP2 FOR ERROR CALL
2367 004632 000402          BR    26$ ;BRANCH TO CALL ERROR
2368 004634 011037 001200          25$: MOV    (R0),$TMP2 ;GET RECEIVED DATA FOR ERROR CALL
2369 004640 010337 001176          26$: MOV    R3,$TMP1 ;GET EXPECTED DATA FOR ERROR CALL
2370 004644 011646          MOV    (SP),-(SP) ;PUT AN EXTRA RETURN ADDRESS ON STACK FOR POSSIBLE LOOP
2371 004646 013737 001310 005764 MOV    UBM24L,EADRES ;LOAD LOWER 16 BITS OF ADDRESS FOR ERROR CALL
2372 004654 013737 001312 005766 MOV    UBM24U,EADRES+2 ;LOAD UPPER 6 BITS OF ADDRESS FOR ERROR CALL
2373 004662 012737 000001 004312 MOV    #1,LOEFLG ;SET FLAG SHOWING AN ERROR RETURN
2374 004670 000207          RTS    PC ;RETURN TO THE ERROR
2375 004672 005227          3$: INC    (PC)+ ;INCREMENT ONE TIME ENTRANCE FLAG
2376 004674 177777          35$: .WORD -1 ;FLAG

```

2377	004676	001006			BNE	5\$:BRANCH IF I'VE BEEN HERE BEFORE
2378	004700	013737	172350	001356	MOV	KIPAR4,RSIZE	:SAVE UPPER LIMIT OF MEMORY
2379	004706	000402			BR	5\$:BRANCH OVER ERROR LOOP CORRECTION
2380	004710	062706	000002		ADD	#2,SP	:POP EXCESS RETURN ADDRESS OFF STACK
2381	004714	062737	000100	001364	ADD	#100,DATA	:CHANGE PATTERN FOR NEXT LOAD
2382	004722	062737	000100	172350	ADD	#100,KIPAR4	:ADD 2K TO PAR4
2383	004730	062777	010000	174342	ADD	#10000,@LREGL	:ADD 2K TO MAP REGISTER
2384	004736	001263			BNE	1\$:BRANCH IF MAP REGISTER NOT ZERO
2385	004740	005277	174336		INC	@LREGU	:ADD ONE TO UPPER 6 BITS OF MAP REG
2386	004744	022777	000073	174330	CMP	#73,@LREGU	:SEE IF TOP 128K BLOCK HAS BEEN PASSED
2387	004752	103255			BHIS	1\$:BRANCH IF NOT PAST IT
2388	004754	005237	001364		INC	DATA	:CHANGE DATA PATTERN FOR NEXT PASS
2389	004760	042737	177700	001364	BIC	#177700,DATA	:CLEAR UPPER 10 BITS OF DATA PATTERN
2390	004766	052737	000300	001364	BIS	#300,DATA	:START WITH 3XX IN DATA PATTERN
2391	004774	005726			TST	(SP)+	:POP STACK EXPOSING FINAL RETURN ADDRESS
2392	004776	000207			RTS	PC	:EXIT

```
2393          .SBTTL  SUBROUTINE TO OBTAIN CONTENTS OF LMA CONTENTS LOCATION
2394 005000 013705 177734          PTMP2: MOV    LMALOW,R5      ;MOVE LMA LOW REGISTER CONTENTS TO R5 FOR SHIFTING
2395 005004 072527 177772          ASH    #-6,R5        ;SHIFT PHYSICAL UPPER 3 BITS TO THE RIGHT 6 PLACES
2396 005010 042705 176177          BIC    #176177,R5    ;CLEAR ALL BUT THE THREE SHIFTED BITS
2397 005014 010546                MOV    R5,-(SP)      ;SAVE THIS NUMBER ON THE STACK
2398 005016 013705 177736          MOV    LMAHI,R5     ;MOVE LMA HIGH REGISTER CONTENTS TO R5 FOR SHIFTING
2399 005022 072527 000012          ASH    #10.,R5      ;SHIFT LOWER 6 BITS TO THE LEFT 10 PLACES
2400 005026 062605                ADD    (SP)+,R5     ;ADD PREVIOUSLY SHIFTED CONTENTS TO R5
2401 005030 013746 172346          MOV    KIPAR3,-(SP) ;SAVE PAR3 ON STACK
2402 005034 010537 172346          MOV    R5,KIPAR3   ;MOVE OBTAINED PAR VALUE TO PAR3
2403 005040 013737 177734 001200  MOV    LMALOW,$TMP2 ;GET ADDRESS MAP CREATED
2404 005046 042737 100000 001200  BIC    #BIT15,$TMP2 ;CLEAR BIT 15 AND
2405 005054 052737 060000 001200  BIS    #60000,$TMP2 ;SET BITS 13 & 14 TO PUT PAR PAGE 3 IN $TMP2
2406 005062 017737 174112 001200  MOV    @$TMP2,$TMP2 ;MOVE CONTENTS OF LOCATION TO $TMP2
2407 005070 012637 172346          MOV    (SP)+,KIPAR3 ;RESTORE PAR3
2408 005074 000207                RTS    PC           ;EXIT
```

2409
 2410
 2411
 2412
 2413
 2414
 2415
 2416
 2417
 2418
 2419
 2420
 2421
 2422
 2423
 2424
 2425
 2426
 2427
 2428
 2429
 2430
 2431
 2432
 2433
 2434
 2435
 2436
 2437
 2438
 2439
 2440

005076 017637 000000 005134
 005104 062716 000002
 005110 017637 000000 005154
 005116 062716 000002
 005122 032777 004000 174006
 005130 001413
 005132 013746
 005134 000000
 005136 042716 177700
 005142 023726 001312
 005146 001004
 005150 023737 001310
 005154 000000
 005156 001402
 005160 062716 000002
 005164 000207

```

.SBTTL SUBROUTINE TO CHECK FOR 11/24 WITH UBMEMORY
*****
*
* THIS SUBROUTINE CHECKS THE SWITCH REGISTER TO SEE IF USER STATES THAT
* THIS IS AN 11/24 CPU WITH UNIBUS MEMORY ONLY. IF NOT, AN EXIT IS
* EXECUTED WITH THE RETURN BEING UPDATED TO THE 2ND LOCATION AFTER THE
* JSR CALL. IF IT IS, THE LMA HIGH REGISTER IS CHECKED FOR BEING EQUAL
* TO THE CONTENTS OF UBM24U, (ASSUMED TO BE PRELOADED. THESE 6 BITS ARE
* THE UPPER 6 BITS OF THE MAPPED ADDRESS FORMED BY THE MAP REGISTER
* LOGIC). IF NOT, EXIT TO FUDGE RETURN. IF SO, THE LMA LOW REGISTER IS
* CHECKED. IT IS ASSUMED THAT LOCATION UBM24L CONTAINS THE EXPECTED
* PHYSICAL ADDRESS. IT COMPARES THE LMA LOW REGISTER WITH UBM24L, AND IF
* EQUAL, EXECUTES A RETURN WITHOUT THE FUDGING OF THE RETURN ADDRESS,
* OTHERWISE IT IS FUDGED.
*
*****
CHKLMA: MOV @0(SP),10$ :MOVE LMAHI ADDRESS TO ACCESS TO 10$
        ADD #2,(SP) :ADVANCE TO NEXT PARAMETER
        MOV @0(SP),11$ :MOVE LMALOW ADDRESS TO ACCESS TO 11$
        ADD #2,(SP) :CORRECT RETURN OVER PARAMETER
        BIT #BIT11,@SWR :SEE IF USER SAYS THIS IS AN 11/24 WITH UB MEMORY
        BEQ 1$ :BRANCH OUT IF NOT
        MOV @(PC)+,-(SP) :MOVE LMA HIGH REGISTER CONTENTS TO STACK
10$: .WORD 0 :LOCATION FOR ADDRESS TO ACCESS
        BIC #177700,(SP) :CLEAR ALL BUT LOWER 6 BITS (UPPER 6 BITS OF ADDRESS)
        CMP UBM24U,(SP)+ :SEE IF UPPER 6 BITS ARE AS EXPECTED
        BNE 1$ :BRANCH TO PREPARE FOR 2ND RETURN LOCATION IF NOT
        CMP UBM24L,@(PC)+ :SEE IF EXPECTED DATA WAS CLOKED PROPERLY
11$: .WORD 0 :LOCATION FOR ADDRESS TO ACCESS
        BEQ 2$ :BRANCH AROUND STACK RETURN FUDGE IF OK
        ADD #2,(SP) :FUDGE RETURN OVER NON-ERROR BRANCH
2$: RTS PC :EXIT
    
```

```
2441                                     .SBTTL  PRETEST DATA SETUP SUBROUTINE
2442                                     :*****
2443 005166 011600      PRETST: MOV      (SP),R0          ;MOVE RETURN ADDRESS TO R0
2444 005170 012037 001362      MOV      (R0)+,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST FOR ESCAPE ON PAR ERRORS
2445 005174 012037 001106      MOV      (R0)+,$LPERR      ;SET LOOP ON ERROR POINTER TO 20$ IN TEST
2446 005200 012037 001100      MOV      (R0)+,$STSTM      ;SETUP TEST NUMBER AND CLEAR THE ERROR FLAG
2447 005204 013777 001100 173726  MOV      $STNM,@DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
2448 005212 010016      MOV      R0,(SP)          ;FUDGE RETURN OVER PARAMETERS
2449 005214 010037 001104      MOV      R0,$LPADR      ;SET LOOP ON TEST POINTER TO START OF TEST
2450 005220 012737 000001 001212  MOV      #1,$TIMES      ;RESET ITERATIONS COUNTER TO 1
2451 005226 000207      RTS      PC          ;RETURN TO BEGIN TEST
```

2452
2453
2454
2455
2456
2457
2458
2459 005230 105737 020162
2460 005234 001411
2461 005236 105737 020163
2462 005242 100006
2463 005244 033715 005774
2464 005250 001403
2465 005252 011537 001176
2466 005256 000402
2467 005260 062716 000002
2468 005264 000207

```
.SBTTL  DISABLE CHECK SUBROUTINE
:*****
:*****
:
:*      THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
:*      IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
:*      MINES IF THE LOCATION SHOULD BE DISABLED.
DSABLD: TSTB  $ENV          ;TEST APT STATUS
        BEQ   1$           ;BRANCH IF NOT APT
        TSTB  $ENVM        ;DOES APT SAY TO SIZE
        BPL   1$           ;BRANCH TO EXIT IF NOT
        BIT   FLOATR,(R5)  ;TEST DISABLE STATUS
        BEQ   1$           ;BRANCH IF IT SHOULD BE DISABLED
        MOV   (R5),$TMP1   ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
        BR    2$           ;BRANCH OVER RETURN CORRECTION
1$:     ADD   #2,(SP)      ;CHOCOLATE FUDGE RETURN OVER ERROR CALL
2$:     RTS   PC          ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
```


2469
2470
2471
2472
2473
2474 005266 105737 020162
2475 005272 001411
2476 005274 105737 020163
2477 005300 100006
2478 005302 033715 005774
2479 005306 001003
2480 005310 011537 001176
2481 005314 000402
2482 005316 062716 000002
2483 005322 000207

```
.SBTTL  ENABLE CHECK SUBROUTINE
:*****
:*      THIS SUBROUTINE CHECKS THE STATUS OF THE BIT POINTED TO IN FLOATR
:*      IN THE LOCATION POINTED TO BY R5 (EITHER $DDW0 OR $DDW1), AND DETER-
:*      MINES IF THE LOCATION SHOULD BE ENABLED.
ENABLD: TSTB  $ENV          ;TEST APT STATUS
        BEQ   1$          ;BRANCH IF NOT APT
        TSTB $ENVM        ;DOES APT SAY TO SIZE
        BPL   1$          ;BRANCH TO EXIT IF NOT
        BIT   FLOATR,(R5) ;TEST DISABLE STATUS
        BNE   1$          ;BRANCH IF IT SHOULD BE ENABLED
        MOV  (R5),$TMP1   ;MOVE CONTENTS OF DEVICE DESCRIPTOR WORD TO $TMP1
        BR   2$          ;BRANCH OVER RETURN CORRECTION
1$:     ADD  #2,(SP)      ;VANILLA FUDGE RETURN OVER ERROR CALL
2$:     RTS  PC          ;RETURN WITHOUT CORRECTING STACK SO ERROR WILL CALL
```

```

2484      .SBTTL  CACHE TEST IN SUBROUTINE FORM
2485      ;;*****
2486      .ENABL  LSB
2487 005324 042737 000001 177572 CASHSR: BIC  #BIT00,MMRO ;TURN OFF RELOCATION
2488 005332 052737 000400 177746      BIS  #BIT08,CACHE ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
2489 005340 032737 010000 177746 1$:  BIT  #BIT12,CACHE ;WAIT TILL DONE
2490 005346 001374
2491 005350 013702 000000      MOV  0,R2 ;SAVE ADDR. 0 CONTENTS
2492 005354 005037 000000      CLR  0 ;0'S TO MAIN MEMORY LOCATION 0.
2493 005360 005003      CLR  R3 ;CLEAR ERROR FLAG
2494 005362 012704 177777      MOV  #-1,R4 ;ALL 1'S TO R4
2495 005366 013700 000114      MOV  CTRAPV,R0 ;SAVE VECTORS
2496 005372 013701 000116      MOV  CTRAPS,R1
2497 005376 012737 005524 000114      MOV  #3$,CTRAPV ;SETUP FOR CACHE TRAP
2498 005404 012737 000340 000116      MOV  #340,CTRAPS
2499 005412 112737 000002 177750      MOVB #2,MAINT ;HODO ALLOWS CACHE UPDATES AND CLOCKING OF
2500      ;PARITY INFO TO INTERRUPT LOGIC ONLY DURING
2501      ;THE DESTINATION ACCESS OF AN INSTRUCTION.
2502 005420 012737 000015 177746      MOV  #15,CACHE ;NO UCB SO AS TO WRITE CACHE STORES
2503 005426 005737 040000      TST  40000 ;UPDATE CACHE LOCATION 0000 WITH CORRECT
2504      ;PARITY STORAGE
2505 005432 052737 000100 177746      BIS  #BIT06,CACHE ;ALLOW WRITE WRONG PARITY DATA TO LO & HI BYTE
2506      ;PARITY STORE.
2507 005440 005737 000000      TST  0 ;READ UPDATE TO CACHE LOCATION 0000;
2508      ;WRITE WRONG PARITY TO HI/LO BYTE PARITY STORES
2509 005444 042737 000100 177746      BIC  #BIT06,CACHE ;DISABLE WWP
2510 005452 005037 177744      CLR  CMPE ;CLEAR CMPE AND PARITY DETECT LOGIC
2511 005456 142737      BICB (PC)+,a(PC)+ ;ALLOW INT & ENABLE LOW CACHE (1ST ;DPM002
2512 005460 000000      CASH1: .WORD 0 ;LOCATION LOADED BY THE TEST ;DPM002
2513 005462 177746      .WORD CACHE ;ADDRESS OF CACHE LOCATION ;DPM002
2514 005464 122727      CMPB (PC)+,(PC)+ ;WHICH TEST ;DPM002
2515 005466 000002      .WORD 2 ;TEST 2 IS TESTED FOR ;DPM002
2516 005470 000000      CASH2: .WORD 0 ;THIS LOCATION LOADED BY THE TEST ;DPM002
2517 005472 001405      BEQ  2$ ;GO TO 2ND TEST SECTION IF 2ND TEST EXECUTING
2518 005474 005737 000000      TST  0 ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS
2519      ;WILL DETECT WRONG PARITY AND THE PARITY
2520      ;ERROR WILL BE CLOCKED TO INTERRUPT LOGIC
2521 005500 000240      NOP ;NEEDED FOR 11/44
2522 005502 005203      INC  R3 ;INDICATE THAT TRAP DID NOT OCCUR
2523 005504 000410      BR   4$ ;BRANCH OVER STACK CORRECTION AND 2ND TEST SECTION
2524 005506 052737 000200 177746 2$:  BIS  #BIT07,CACHE ;ALLOW FOR ABORT
2525 005514 011304      MOV  (R3),R4 ;READ HIT LO & HI BYTE PARITY CHECK GENERATORS WILL
2526      ;DETECT WRONG PARITY USING HODO AND SOURCE MODE FOR READING LOCATION 0 WILL
2527      ;INHIBIT PARITY ERROR FROM BEING CLOCKED TO INTERRUPT LOGIC. HOWEVER, THE PARITY
2528      ;ERROR SIGNAL WILL CAUSE THE ABORT SIGNAL TO BE ASSERTED. THE ABORT SIGNAL WILL
2529      ;BECAUSE CMPE<15> TO BE SET. THIS INSTRUCTION SHOULD BE ABORTED
2530 005516 000240      NOP ;NEEDED IN AN 11/44 TO ALLOW 1 INSTRUCTION BEFORE ABORT
2531 005520 005203      INC  R3 ;INDICATE NO TRAP OCCURED
2532 005522 000401      BR   4$ ;BRANCH OVER STACK CORRECTION
2533 005524 022626      3$:  CMP  (R6)+,(R6)+ ;READJUST STACK DUE TO INTERRUPT
2534 005526 005037 177744      4$:  CLR  CMPE ;CLEAR CMPE
2535 005532 012737 001015 177746      MOV  #1015,CACHE ;DISABLE CACHE
2536 005540 105037 177750      CLRB MAINT ;DISABLE MAINT. MODE
2537 005544 010237 000000      MOV  R2,0 ;RESTORE LOCATION 0
2538 005550 010037 000114      MOV  R0,CTRAPV ;RESTORE CACHE INTERRUPT VECTORS
2539 005554 010137 000116      MOV  R1,CTRAPS
2540 005560 052737 000400 177746      BIS  #BIT08,CACHE ;BEFORE LEAVING TEST FLUSH CACHE TO ELIMINATE EFFECTS OF WWP

```

```
2541 005566 032737 010000 177746 5$: BIT #BIT12,CACHE ;WAIT TILL DONE
2542 005574 001374 BNE 5$
2543 005576 000207 RTS PC ;EXIT
2544 .DSABL LSB
2545 .SBTTL SUBROUTINE TO PREPARE AND CHECK DATA PATTERN
2546 ::*****
2547 005600 011504 CHKPAT: MOV (R5),R4 ;MOVE NEXT COUNT PATTERN TO R2
2548 005602 043704 005776 BIC MASK1,R4 ;USE MASK1 PRE-LOADED FOR PROPER LOADING
2549 005606 010410 MOV R4,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
2550 005610 011504 MOV (R5),R4 ;RELOAD PATTERN
2551 005612 043704 006000 BIC MASK2,R4 ;USE THE 2ND MASK TO CONSTRUCT EXPECTED VALUE
2552 005616 020410 CMP R4,(R0) ;COMPARE EXPECTED WITH RECEIVED
2553 005620 001402 BEQ 1$ ;BRANCH IF DATA IS OK
2554 005622 062716 000002 ADD #2,(SP) ;FUDGE RETURN TO SHOW ERROR
2555 005626 000207 1$: RTS PC ;EXIT
```

```

2556                                     .SBTTL READ AN OCTAL NUMBER FROM THE TTY
2557                                     :*****
2558                                     :*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY, CHANGE IT TO
2559                                     :*BINARY AND PUT THE NUMBER ON THE STACK.
2560                                     :*CALL:
2561                                     :*   RDOCT      ;READ AN OCTAL NUMBER
2562                                     :*   MOV      (SP)+,LOCATION ;POP THE INPUTED NUMBER OFF STACK
2563                                     :*                                     ;HIGH ORDER BITS ARE IN $HIOCT
2564
2565 005630 011646 $RDOCT: MOV      (SP),-(SP) ;MAKE ROOM FOR THE
2566 005632 016666 000004 000002 MOV      4(SP),2(SP) ;OCTAL NUMBER
2567 005640 010046 MOV      R0,-(SP) ;SAVE R0
2568 005642 010146 MOV      R1,-(SP) ;SAVE R1
2569 005644 010246 MOV      R2,-(SP) ;SAVE R2
2570 005646 104407 1$: RDLIN ;READ THE OCTAL NUMBER
2571 005650 012600 MOV      (SP)+,R0 ;GET ADDRESS OF 1ST CHARACTER
2572 005652 005001 CLR      R1 ;CLEAR R1
2573 005654 005002 CLR      R2 ;CLEAR R2
2574 005656 112046 2$: MOVVB   (R0)+,-(SP) ;MOVE CHARACTER TO STACK
2575 005660 001415 BEQ      3$ ;IF ZERO, EXIT
2576 005662 000241 CLC ;CLEAR THE CARRY BIT
2577 005664 006101 ROL      R1 ;SHIFT MSB TO THE CARRY BIT
2578 005666 006102 ROL      R2 ;SHIFT IT TO UPPER RECEIVER
2579 005670 000241 CLC ;CLEAR THE CARRY BIT
2580 005672 006101 ROL      R1 ;SHIFT MSB TO THE CARRY BIT
2581 005674 006102 ROL      R2 ;SHIFT IT TO UPPER RECEIVER
2582 005676 000241 CLC ;CLEAR THE CARRY BIT
2583 005700 006101 ROL      R1 ;SHIFT MSB TO THE CARRY BIT
2584 005702 006102 ROL      R2 ;SHIFT IT TO UPPER RECEIVER
2585 005704 042716 177770 BIC      #177770,(SP) ;STRIP ALL BUT BINARY EQUIVALENT
2586 005710 052601 BIS      (SP)+,R1 ;SET THE BINARY EQUIVALENT TO LOWER RECEIVER
2587 005712 000761 BR      2$ ;BRANCH BACK
2588 005714 005726 3$: TST      (SP)+ ;CLEAN TERMINATOR FROM STACK
2589 005716 010166 000012 MOV      R1,12(SP) ;SAVE THE RESULT
2590 005722 010237 005736 MOV      R2,$HIOCT
2591 005726 012602 MOV      (SP)+,R2 ;RESTORE R2
2592 005730 012601 MOV      (SP)+,R1 ;RESTORE R1
2593 005732 012600 MOV      (SP)+,R0 ;RESTORE R0
2594 005734 000002 RTI ;RETURN
2595 005736 000000 $HIOCT: .WORD 0 ;HIGH ORDER BITS GO HERE
  
```

DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC

2596					.SBTTL	DATA TABLES AND ASCII STRINGS USED IN THIS DIAGNOSTIC
2597	005740	005744	006120		DTMSG: .WORD	DTMSG,DFMSG ; POINTER TO DATA VARIABLE ADDRESSES
2598	005744	001266	001270	001272	DTMSG: .WORD	MMRLOW,MMRHI,UBRLOW,UBRHI,\$TESTN,0 ; DATA VARIABLES TO BE PRINTED
2599	005760	000000			CHARCT: .WORD	0 ; THIS LOCATION HOLDS CHARACTERS INPUTED DURING THE TYPE ROUTINE
2600	005762	000000			EXTOUT: .WORD	0 ; THIS LOCATION STORES THE OUTPUT OF THE EXTRACTION ROUTINE
2601	005764	000000	000077		EADRES: .WORD	0,77 ; LOCATIONS FOR STORING 22 BITS OF THE UBMAR REGISTER ADDRESS
2602	005770	000000	000077		EADRS2: .WORD	0,77 ; LOCATIONS FOR STORING ANOTHER UBMAR REGISTER ADDRESS
2603	005774	000000			FLOATR: .WORD	0 ; LOCATION TO HOLD BIT TO FLOAT TO TEST DDW'S STATUS
2604	005776	000000			MASK1: .WORD	0 ; PRE-LOADED BY THE TEST WITH THE 1ST BIT-MASK
2605	006000	000000			MASK2: .WORD	0 ; PRE-LOADED BY THE TEST WITH THE 2ND BIT-MASK
2606	006002				SPECST: .BLKW	40
2607		000002			.RADIX	2 ; SET RADIX TO BINARY REPRESENTATION
2608	006102	177777			PATRNS: .WORD	1111111111111111 ; ALL BITS SET
2609	006104	000000			.WORD	0000000000000000 ; ALL BITS CLEAR
2610	006106	125252			.WORD	1010101010101010 ; ODD BITS SET, EVEN BITS CLEAR
2611	006110	052525			.WORD	0101010101010101 ; EVEN BITS SET, ODD BITS CLEAR
2612	006112	031463			.WORD	0011001100110011 ; ALTERNATING PAIRS OF BITS SET
2613	006114	007417			.WORD	0000111100001111 ; ALTERNATING GROUPS OF 4 BITS SET
2614	006116	000377			.WORD	0000000011111111 ; LOWER BYTE SET, UPPER BYTE CLEAR
2615		000010			.RADIX	8 ; SET RADIX BACK TO OCTAL
2616	006120	000	000	000	DFMSG: .BYTE	0,0,0,0,0 ; ALL NUMBERS ARE TO BE PRINTED IN OCTAL
2617	006125	200	125	116	UBMAVA: .ASCIZ	<CRLF>?UNIBUS MEMORY AVAILABLE = ?
2618	006161	123	127	122	NEWSWR: .ASCIZ	?SWR INPUT ?
2619	006174	040	113	200	UBMEND: .ASCIZ	? K?<CRLF>


```

2636          .SBTTL  PRE-TESTING SETUP
2637          :*****
2638          :START OF TEST CODE
2639          :*****
2640
2641          010000          .=10000          ;START TEST CODE AT ADDRESS 10000 (2K)
2642 010000 012737 000340 177776 START: MOV #340,PS          ;LOCK OUT ALL INTERRUPTS
2643 010006 012700 001100          MOV #SCMTAG,R0          ;FIRST LOCATION TO BE CLEARED
2644 010012 012701 000021          MOV #<STKS-SCMTAG>/2,R1;MOVE LOOP COUNTER TO R1
2645 010016 005020          11$: CLR (R0)+          ;CLEAR MEMORY LOCATION
2646 010020 077102          SOB R1,11$          ;SUBTRACT 1 AND BRANCH IF NOT DONE YET
2647 010022 005037 020150          CLR $PASS          ;INITIALIZE PASS COUNT
2648 010026 012706 001100          MOV #STACK,SP          ;INITIALIZE STACK
2649 010032 012700 010054          MOV #13$,R0          ;MOVE ADDRESS OF VECTORS TO R0
2650 010036 012701 000005          MOV #5,R1          ;DO 5 LOADS
2651 010042 012030          12$: MOV (R0)+,@(R0)+          ;MOVE VECTOR TO LOCATION
2652 010044 012730 000340          MOV #340,@(R0)+          ;MOVE PRIORITY 7 TO NEXT LOCATION
2653 010050 077104          SOB R1,12$          ;BRANCH BACK IF NOT DONE YET
2654 010052 000417          BR 14$          ;BRANCH OVER DATA WORDS
2655 010054 020266 000020 000022 13$: .WORD $SCOPE,IOTVEC,IOTVEC+2,$ERROR,EMTVEC,EMTVEC+2,$TRAP,TRAPVE
2656 010074 000036 023610 000024          .WORD TRAPVE+2,$PWRDN,PWRVEC,PWRVEC+2,$RTRN,TBITVE,TBITVE+2
2657 010112 013737 021644 021636 14$: MOV $ENDCT,$EOPCT          ;SETUP END-OF-PROGRAM COUNTER
2658 010120 005037 001214          CLR $ESCAPE          ;CLEAR THE ESCAPE ON ERROR ADDRESS
2659 010124 112737 000001 001113          MOV #1,$ERMAX          ;ALLOW ONE ERROR PER TEST
2660 010132 012737 000002 022070          MOV #RTI,$RTRN          ;SET $RTRN TO AN RTI
2661 010140 012737 010166 000010          MOV #16$,RESVEC          ;MOVE 16$ (FAILURE) TO RESVEC
2662 010146 012746 000340          MOV #340,-(SP)          ;MOVE PRIORITY 7 TO STACK
2663 010152 012746 010160          MOV #15$,-(SP)          ;MOVE 15$ (SUCCESS) TO THE STACK
2664 010156 000006          RTT          ;TRY TO DO AN RTT
2665 010160 012737 000006 022070 15$: MOV #RTT,$RTRN          ;RTT IS LEGAL--SET $RTRN TO AN RTT
2666 010166 012706 001100          16$: MOV #STACK,SP          ;RESET STACK IF NECESSARY
2667 010172 012737 000012 000010          MOV #RESVEC+2,RESVEC;RESET TRAP CATCHER
2668 010200 005037 022076          CLR $TBIT          ;CLEAR "T" BIT SWITCH
2669 010204 012737 010700 001104          MOV #TST1+2,$LPADR          ;SETUP $LPADR
2670 010212 012737 010700 001106          MOV #TST1+2,$LPERR          ;SETUP $LPERR
2671 010220 005227 177777          INC #-1          ;FIRST TIME?
2672 010224 001017          BNE 18$          ;BRANCH IF NOT
2673 010226 104401 010234          TYPE ,17$          ;TYPE THE TEST TITLE
2674 010232 000414          BR 18$          ;BRANCH OVER ASCIZ
2675 010234 103 113 113 17$: .ASCIZ 'CKKUAEO 11/24/44 UBI MAP'
2676          .EVEN
2677 010264 005737 020150          18$: TST $PASS          ;IS THIS THE FIRST PASS?
2678 010270 001140          BNE 8$          ;BRANCH IF NOT
2679 010272 013746 000004          MOV 4,-(SP)          ;SAVE TIMEOUT VECTOR
2680 010276 012737 010470 000004          MOV #2$,4          ;TIMEOUTS TO 104$
2681 010304 013746 000006          MOV 6,-(SP)          ;SAVE PS VECTOR
2682 010310 012737 000340 000006          MOV #340,6          ;PRIORITY 7
2683 010316 000007          MFPT          ;DETERMINE PROCESSOR TYPE
2684 010320 110037 007140          MOV #R0,CPUTYP          ;MOVE THE CPU NUMBER TO CPUTYP
2685 010324 105337 007140          DECB CPUTYP          ;MAKE THE 11/44 VALUE ZERO
2686 010330 001021          BNE 1$          ;BRANCH TO NEXT TEST IF NOT AN 11/44
2687 010332 022737 177777 177570          CMP #-1,177570          ;SEE IF SSWR IS TO BE USED          :DPM002
2688 010340 001004          BNE 1000$          ;BRANCH TO SETUP FOR HARDWARE SWR          :DPM002
2689 010342 012737 000176 001136          MOV #176,SWR          ;SET UP FOR SSWR          :DPM002
2690 010350 000403          BR 1010$          ;BRANCH          :DPM002
2691 010352 012737 177570 001136 1000$: MOV #177570,SWR          ;SET UP FOR HARDWARE SWITCH REGISTER          :DPM002
2692 010360 112737 000064 006255 1010$: MOV #4,CPUMSG+55          ;MOVE ASCII 4 TO LOCATION IN MESSAGE TO PRINT

```

```

2693 010366 104401 006200          TYPE      ,CPUMSG      ;TYPE THE CPU TYPE HEADER
2694 010372 000446          BR          6$        ;CONTINUE
2695 010374 122737 000002 007140 1$:  CMPB      #2,CPUTYP  ;SEE IF THIS IS AN 11/24
2696 010402 001034          BNE        3$        ;BRANCH TO FATAL ERROR MESSAGE PRINTING IF NOT
2697 010404 012737 000176 001136  MOV      #$$SWR,SWR  ;SETUP SOFTWARE SWITCH REGISTER ;DPM002
2698 010412 112737 000062 006255  MOVVB   #'2,CPUMSG+55 ;MOVE ASCII 2 TO LOCATION IN MESSAGE TO PRINT
2699 010420 104401 006200          TYPE      ,CPUMSG      ;TYPE THE CPU TYPE HEADER
2700 010424 105737 020163          TSTB     $ENVM      ;IS APT SIZING
2701 010430 100427          BMI          6$        ;BRANCH IF NO
2702 010432 012737 000176 001136  MOV      #$$SWR,SWR  ;SETUP SOFTWARE SWITCH REGISTER
2703 010440 005737 000176          TST      $$SWR      ;SEE IF SWR IS NON-ZERO
2704 010444 001021          BNE        6$        ;BRANCH IF SO
2705 010446 104401 006161          TYPE     ,NEWSWR    ;TYPE: 'SWR INPUT '
2706 010452 104410          RDOCT     ;GO READ USER OCTAL INPUT
2707 010454 012637 000176          MOV      (SP)+,$$SWR ;MOVE NEW CONTENTS TO THE SOFTWARE SWR LOCATION
2708 010460 001013          BNE        6$        ;BRANCH IF NON-ZERO
2709 010462 005237 000176          INC      $$SWR      ;MAKE SWR NON-ZERO FOR POSSIBLE NEXT RUN
2710 010466 000410          BR          6$        ;CONTINUE
2711 010470 062706 000004          2$: ADD     #4,SP      ;CLEAN STACK AFTER TIMEOUT
2712 010474 005237 020142          3$: INC     $MSGTY   ;TELL APT THIS IS A FATAL ERROR
2713 010500 104401 006607          4$: TYPE   ,BADCPU   ;TYPE THE BAD CPU MESSAGE
2714 010504 000000          HALT     ;FATAL ERROR - THIS DIAGNOSTIC IS WRITTEN
2715                                ;FOR 11/24 AND 11/44 PROCESSORS ONLY
2716 010506 000774          BR          4$        ;DON'T ALLOW CONTINUE
2717 010510 012637 000006          6$: MOV     (SP)+,6    ;RESTORE TIMEOUT PS
2718 010514 012637 000004          MOV     (SP)+,4    ;RESTORE TIMEOUT VECTOR
2719 010520 013746 172346          MOV     KIPAR3,-(SP) ;SAVE PAR3
2720 010524 012737 000400 172346  MOV     #400,KIPAR3  ;START WITH ADDRESS 40000
2721 010532 012704 060000          MOV     #60000,R4   ;ADDRESS PAR3, OFFSET=0
2722 010536 012705 040222          MOV     #40222,R5   ;DATA TO LOAD TO 40000
2723 010542 010514          7$: MOV     R5,(R4)    ;MOVE DATA TO LOCATION
2724 010544 062705 010000          ADD     #10000,R5   ;MOVE DATA UP 10000
2725 010550 062737 000100 172346  ADD     #100,KIPAR3  ;MOVE ADDRESS UP 10000
2726 010556 022737 007600 172346  CMP     #7600,KIPAR3 ;SEE IF ALL DONE YET
2727 010564 001366          BNE      7$        ;BRANCH BACK IF NOT
2728 010566 012637 172346          MOV     (SP)+,KIPAR3 ;RESTORE PAR3
2729 010572 105737 020163          8$: TSTB   $ENVM      ;IS APT SIZING
2730 010576 100003          BPL      LOOP      ;BRANCH IF NOT
2731 010600 012737 020164 001136  MOV     #$$SWREG,SWR ;USE APT SWITCH REGISTER
2732 010606 012737 003372 000250 LOOP: MOV     #MMTRAP,MMVEC ;LOAD MEMORY MANAGEMENT TRAP SERVICE ROUTINE ADDRESS
2733 010614 012737 000340 000252  MOV     #340,MMVEC+2 ;SET PRIORITY SEVEN
2734 010622 012737 003222 000004  MOV     #CPUER,ERRVEC ;LOAD CPU TRAP SERVICE ROUTINE ADDR
2735 010630 012737 000340 000006  MOV     #340,ERRVEC+2 ;SET PRIORITY SEVEN
2736 010636 005037 001326          CLR     CPUEXP      ;NOT EXPECTING ANY CPU ERRORS
2737 010642 005037 177572          CLR     MMRO        ;START IN 16 BIT MAPPING
2738 010646 005037 172516          CLR     MMR3        ;DISABLE MAP AND 22-BIT MAPPING
2739 010652 012700 177777          MOV     #-1,R0      ;NEGATIVE ONE USED TO INITIALIZE FLAGS
2740 010656 010037 003224          MOV     R0,CPFLAG   ;INITIALIZE FLAGS
2741 010662 010037 003032          MOV     R0,TOFLAG   ;INITIALIZE FLAGS
2742 010666 010037 003374          MOV     R0,MMFLAG   ;INITIALIZE FLAGS
2743 010672 010037 177766          MOV     R0,CPUERR   ;CLEAR CPU ERROR REGISTER

```


2753

```
.SBTTL TEST # 1 - MAP REGISTER RESPONSE TEST
:*****
:*TEST 1      MAP REGISTER RESPONSE TEST
:*
:* THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
:* CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
:* THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
:* AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
:* BE GIVEN.
:*
:*****
```

```
TST1:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST2,20$,1 ;DATA USED BY PRETST
MOV #TIMEOUT,ERRVEC ;LOAD ERRVEC WITH ROUTINE ADDRESS
TSTB CPUTYP ;TEST TO SEE WHICH CPU IS RUNNING THIS DIAGNOSTIC
BEQ 3$ ;BRANCH AROUND MAP REGISTER EXISTENCE CHECK IF 11/44
CLR ERRCNT ;CLEAR THE ERROR COUNTER
1$: MOV MAPLO,R2 ;READ FIRST MAP REGISTER TO R2
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ 3$ ;BRANCH TO CHECK THEM ALL IF NONE, THEY ARE IN THIS 11/24
TSTB $ENV ;ARE WE RUNNING UNDER APT
BPL 2$ ;BRANCH IF NOT
INC $MSGTY ;TELL APT THIS IS A FATAL ERROR
HALT ;HALT - FATAL ERROR
BR 1$ ;TRY AGAIN - RESTART REQUESTED
2$: TYPE ,MRQUES ;ASK USER IF THERE ARE MAP REGISTERS IN THIS 11/24
RDCHR ;GO READ USER INPUT
MOVB (SP)+,GMRMD1 ;MOVE THE CHARACTER TO THE PRINTING LOCATION
TYPE ,GMRMD1 ;TYPE THE CHARACTER
CMPB #'Y,GMRMD1 ;SEE IF THIS WAS A 'Y'
BEQ 3$ ;BRANCH AROUND FATAL MESSAGE IF EQUAL TO A 'Y'
TYPE ,GMRMOD ;TYPE: 'DIAGNOSTIC CHECKS THIS MODULE - INSERT
;'BEFORE RE-RUNNING'
;FATAL ERROR - WAIT FOR USER ACTION
BR 1$ ;TRY AGAIN - RESTART REQUESTED
3$: MOV #MAPLO,R0 ;PUT FIRST MAP REGISTER ADDR IN R0
MOV #100,R3 ;TEST ALL MAP REGISTERS
CLR ERRCNT ;CLEAR THE ERROR COUNTER
MOV #20$, $LPERR ;MAKE SURE $LPERR IS POINTING AT 20$
20$: MOV (R0),R2 ;READ MAP REGISTERS TO R2
BIT #BIT09,@SWR ;SEE IF LOOP ON ERROR IS SET
BEQ 4$ ;BRANCH AROUND SETUP IF NOT
TST ERRCNT ;SEE IF AN ERROR OCCURED
BNE 20$ ;BRANCH BACK IF SO
4$: TST (R0)+ ;INCREMENT R0 TO NEXT REGISTER LOCATION
SOB R3,20$ ;DO ALL OF THEM
MOV #CPUER,ERRVEC ;RESTORE CPU TRAP SERVICE ROUTINE ADDRESS TO ERRVEC
MOV #3$, $LPERR ;MOVE 3$ TO LOOP ON ERROR FOR ERROR +4 BELOW
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST2 ;GO TO NEXT TEST IF NO ERRORS
DEC $ERTTL ;DON'T COUNT SUMMARY AS AN ADDITIONAL ERROR
DEC ERRCNT ;DECREMENT ERRCNT FOR SAME REASON
ERROR +4 ;SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ
```

```
010676 000004
010676 004737 005166
010700 011122 011042 000001
2754 010704 012737 003030 000004
2755 010712 105737 007140
2756 010720 001435
2757 010724 005037 001320
2758 010726 013702 170200
2759 010732 005737 001320
2760 010736 001426
2761 010742 105737 020162
2762 010744 100004
2763 010750 005237 020142
2764 010752 000000
2765 010756 000764
2766 010760 104401 006754
2767 010762 104406
2768 010766 112637 007043
2769 010770 104401 007043
2770 010774 122737 000131 007043
2771 011000 001404
2772 011006 104401 007046
2773
2774 011010 000000
2775 011014 000745
2776 011016 012700 170200
2777 011020 012703 000100
2778 011024 005037 001320
2779 011030 012737 011042 001106
2780 011034 011002
2781 011042 032777 001000 170064
2782 011044 001403
2783 011052 005737 001320
2784 011054 001370
2785 011060 005720
2786 011062 077312
2787 011064 012737 003222 000004
2788 011066 012737 011020 001106
2789 011074 005737 001320
2790 011102 001405
2791 011106 005337 001110
2792 011110 005337 001320
2793 011114 104004
```

2808

.SBTTL TEST # 2 - BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS

:TEST 2 BIT PATTERN AND CLEAR TEST OF 40 MAP REGISTERS

 THIS TEST WILL RUN 7 BIT PATTERNS THROUGH BOTH WORDS OF 40 UNIBUS
 MAP REGISTER LOCATIONS MAPL00 - MAPL37, USING TWO MAJOR PASSES.
 IT WILL TEST THE LOWER 16 BITS ON THE FIRST PASS, AND THE UPPER
 6 BITS ON THE SECOND. THIS TEST WILL MAKE SURE THE REGISTER CAN
 CLEAR, AND THEN RUN 6 BIT PATTERNS THROUGH EACH UNIBUS MAP REGISTER.
 IF THE DATA PATTERN RECEIVED DOES NOT MATCH THE EXPECTED PATTERN,
 THEN THE MAP REGISTER ADDRESS, DATA RECEIVED, PATTERN LOADED, AND
 PATTERN EXPECTED ARE REPORTED. AT THE END OF EACH OF THE TWO MAJOR
 PASSES, A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN DETERMINE
 IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.

TST2:

011122	000004								
011122	004737	005166							
011124	011444	011254	000002						
2809	011130	005037	001320						
2810	011142	012737	011254	001106	1\$:				
2811	011150	012700	170200						
2812	011154	005037	005776						
2813	011160	012737	000001	006000					
2814	011166	012701	000040		2\$:				
2815	011172	012702	000007		3\$:				
2816	011176	012705	006102						
2817	011202	004737	005600		4\$:				
2818	011206	000435							
2819	011210	011003							
2820	011212	011537	001174						
2821	011216	011546							
2822	011220	004737	003650						
2823	011224	010346							
2824	011226	004737	003624						
2825	011232	043737	005776	001174					
2826	011240	010046							
2827	011242	013746	172356						
2828	011246	004737	003674						
2829									
2830	011252	000403							
2831	011254	004737	005600		20\$:				
2832	011260	000401							
2833	011262	104203			5\$:				
2834	011264	032777	001000	167644	6\$:				
2835	011272	001370							
2836	011274	012737	011172	001106					
2837	011302	005725			7\$:				
2838	011304	077242							
2839	011306	062700	000004						
2840	011312	077151							
2841	011314	005737	001320						
2842	011320	001416							
2843	011322	012737	011142	001106					
2844	011330	005337	001110						
2845	011334	005337	001320						

```
SCOPE      PC,PRETST       ;GO SET UP PRETEST DATA
JSR          TST3,20$,2     ;DATA USED BY PRETST
WORD        ERRCNT        ;CLEAR THE ERROR COUNT LOCATION
CLR          #20$, $LPERR   ;SETUP 20$ AS LOOP ON ERROR INDICATOR (IF NOT ALREADY)
MOV          #MAPL00,R0     ;MOVE STARTING ADDRESS OF LOWER 16 BITS REGISTER TO R0
CLR          MASK1         ;MOVE '0' TO MASK1
MOV          #1,MASK2      ;MOVE '1' TO MASK2
MOV          #40,R1        ;DO 40 REGISTERS LOOP COUNTER
MOV          #7,R2         ;MOVE PATTERN LOOP COUNTER TO R2
MOV          #PATRNS,R5    ;MOVE PATTERN START TO R5
JSR          PC,CHKPAT     ;GO CHECK PATTERN
BR           7$            ;RETURN IS HERE IF DATA IS OK
MOV          (R0),R3       ;READ BAD MAP REGISTER DATA INTO R3
MOV          (R5), $TMP0   ;MOVE PATTERN TO $TMP0
MOV          (R5),-(SP)    ;MOVE PATTERN TO STACK FOR SUBROUTINE USE
JSR          PC,PATEXT     ;GO SET DATA INTO PATROR AND PATAND
MOV          R3,-(SP)      ;MOVE BAD DATA TO STACK FOR SUBROUTINE USE
JSR          PC,DATEXT     ;SUBROUTINE TO LOAD DATAOR AND DATAND
BIC          MASK1,$TMP0   ;FORM INPUT PATTERN FOR ERROR PRINTING
MOV          R0,-(SP)      ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE
MOV          KIPAR7,-(SP)  ;PUT PAR CONTENTS ON STACK FOR ADREXT SUBROUTINE
JSR          PC,ADREXT    ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
AND          #00000001,ADREXT ;AND FORM A PHYSICAL 22-BIT ADDRESS
BR           5$            ;BRANCH OVER LOOP ON ERROR SECTION
JSR          PC,CHKPAT     ;GO TO SUBROUTINE TO CHECK PATTERN
BR           6$            ;RETURN IS HERE IF DATA IS OK
ERROR       +203          ;THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED
BIT         #BIT9,@SWR    ;SEE IF LOOP ON ERROR IS SET
BNE         20$          ;LOOP BACK IF SO
MOV         #3$, $LPERR   ;MOVE 3$ TO LOOP ON ERROR INDICATOR
TST        (R5)+         ;POINT R5 TO NEXT PATTERN
SOB        R2,4$         ;DECREMENT LOOP COUNTER
ADD        #4,R0         ;POINT TO NEXT MAP REGISTER UNDER TEST
SOB        R1,3$         ;DECREMENT LOOP COUNTER AND
TST        ERRCNT        ;SEE IF THERE WERE ANY ERRORS
BEQ        9$            ;BRANCH IF NO ERRORS
MOV        #1$, $LPERR   ;MOVE 1$ TO LOOP ON ERROR FOR ERROR +6 OR +7 BELOW
DEC        $ERTTL        ;SUMMATION ERROR NOT COUNTED AS ANOTHER ERROR
DEC        ERRCNT        ;SAME AS ABOVE
```

2846	011340	022700	170402			CMP	#MAPH37+4,RO	:SEE IF THIS PASS WAS UPPER 6 BITS
2847	011344	001402				BEQ	8\$:GO SERVICE UPPER 6 BITS ERROR IF SO
2848	011346	104006				ERROR	+6	:SUMMARY OF BIT PATTERN FAILURES, LOWER 16 BITS
2849	011350	000405				BR	10\$:GO SET UP DATA FOR TESTING UPPER 6 BITS
2850	011352	104007			8\$:	ERROR	+7	:SUMMARY OF BIT PATTERN FAILURES, UPPER 6 BITS
2851	011354	000433				BR	TST3	::GO TO NEXT TEST - LOWER 16 BITS ALREADY TESTED
2852	011356	022700	170402		9\$:	CMP	#MAPH37+4,RO	:SEE IF THIS PASS WAS FOR THE UPPER 6 BITS
2853	011362	001430				BEQ	TST3	::GO TO NEXT TEST IF IT WAS
2854	011364	012737	011254	001106	10\$:	MOV	#20\$,SLPERR	:RESET LOOP ON ERROR TO 20\$
2855	011372	012700	170202			MOV	#MAPH00,RO	:MOVE STARTING ADDRESS OF UPPER 6 BITS REGISTER TO RO
2856	011376	012737	177700	005776		MOV	#177700,MASK1	:MOVE 1ST MASK TO MASK1
2857	011404	012737	177700	006000		MOV	#177700,MASK2	:MOVE 2ND MASK TO MASK2
2858	011412	005037	001254			CLR	PATTOR	:CLEAR PATTOR FOR NEXT PASS
2859	011416	005037	001246			CLR	DATAOR	:CLEAR DATAOR FOR NEXT PASS
2860	011422	012737	177777	001252		MOV	#-1,PATAND	:MOVE -1 TO PATAND FOR NEXT PASS
2861	011430	012737	177777	001242		MOV	#-1,DATAND	:MOVE -1 TO DATAND FOR NEXT PASS
2862	011436	005037	001320			CLR	ERRCNT	:CLEAR ERROR COUNT FOR NEXT PASS
2863	011442	000651				BR	2\$:GO REACCOMPLISH TEST FOR UPPER 6 BITS

2876

```
.SBTTL TEST # 3 - DUAL ADDRESS LOADS & READS MAP REG'S
*****
*TEST 3 DUAL ADDRESS LOADS & READS MAP REG'S
*
* THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED
* DURING A 'MOV #DATA,MAPREG' INSTRUCTION. ALL MAP REGISTERS
* ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPL00,
* IS LOADED WITH A -1. THEN, ALL MAP REGISTERS ARE READ,
* STARTING WITH MAPH37, AND VERIFIED TO BE ZERO. ANY REGISTER
* THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT
* OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR. AT THE
* END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.
*****
```

```
TST3:
011444 011444 000004          SCOPE
011446 004737 005166      JSR    PC,PRETST      ;GO SET UP PRETEST DATA
011452 011762 011616 000003 .WORD  TST4,20$,3     ;DATA USED BY PRETST
2877 011460 042737 000001 177572 BIC    #1,MMR0        ;TURN OFF MEMORY MANAGEMENT
2878 011466 012700 170200      MOV    #MAPL00,R0     ;LOAD ADDRESS OF MAPL00 IN R0
2879 011472 012702 177700      MOV    #177700,R2     ;SET UP XOR DATA
2880 011476 012701 170400 1$:  MOV    #MAPH37+2,R1   ;PUT ADDRESS OF MAPH37+2 IN R1
2881 011502 012737 000077 001174 MOV    #77,$TMP0      ;SET UP TEST PATTERN IN $TMP0
2882 011510 004737 002772      JSR    PC,CLRMAP      ;CLEAR ALL MAP REGISTERS FOR TEST
2883 011514 012703 000100      MOV    #100,R3        ;SET UP LOOP COUNTER FOR TESTING 40 REGISTERS, 2 WORDS EACH
2884 011520 005037 172516      CLR    MMR3           ;CLEAR MMR3
2885 011524 012704 000100 2$:  MOV    #100,R4        ;SET UP LOOP COUNTER FOR CHECKING 40 REGISTERS, 2 WORDS EACH
2886 011530 074237 001174      XOR    R2,$TMP0       ;REVERSE BIT STATES OF $TMP0 IN BITS 7 TO 15
2887 011534 013710 001174      MOV    $TMP0,(R0)     ;LOAD MAP REGISTER UNDER TEST
2888 011540 012701 170400      MOV    #MAPH37+2,R1-  ;PUT ADDRESS OF MAPH37+2 IN R1
2889 011544 005741 3$:  TST    -(R1)          ;SEE IF MAP REGISTER IS ZERO
2890 011546 001435      BEQ    7$             ;GO SEE IF MORE REGISTERS TO TEST IF = 0
2891 011550 005011      CLR    (R1)          ;CLEAR THE FAILED LOCATION
2892 011552 020100      CMP    R1,R0          ;SEE IF NON-ZERO REGISTER ADDRESS MATCHES ADDRESS LOADED
2893 011554 001432      BEQ    7$             ;GO SEE IF MORE REGISTERS TO TEST IF SO
2894 011556 012137 001202      MOV    (R1)+,$TMP3    ;MOVE FAULTY DATA TO $TMP3 & PREPARE R1 FOR POSSIBLE LOOP
2895 011562 013746 001202      MOV    $TMP3,-(SP)    ;MOVE FAULTY DATA TO STACK FOR SUBROUTINE USE
2896 011566 013746 172350      MOV    KIPAR4,-(SP)  ;MOVE PAR4 TO STACK FOR SUBROUTINE USE
2897 011572 004737 003674      JSR    PC,ADREXT     ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2898
2899 011576 013746 172356      MOV    KIPAR7,-(SP)  ;PUT PAR ON STACK FOR DATEXT SUBROUTINE USE
2900 011602 010146      MOV    R1,-(SP)      ;PUT VIRTUAL ADDRESS ON STACK FOR DATEXT SUBROUTINE USE
2901 011604 004737 003624      JSR    PC,DATEXT     ;SUBROUTINE TO LOAD DATAOR AND DATAND
2902 011610 010037 005770      MOV    R0,EADRS2     ;MOVE ADDRESS IN R0 TO EADRS2 FOR ERROR CALL
2903 011614 000405      BR     5$            ;BRANCH OVER LOOP ON ERROR SECTION
2904 011616 013710 001174 20$: MOV    $TMP0,(R0)     ;LOAD MAP REGISTER UNDER TEST
2905 011622 005711      TST    (R1)          ;SEE IF MAP REGISTER IS ZERO
2906 011624 001402      BEQ    6$            ;GO SEE IF LOOP ON ERROR IS STILL SET IF ZERO
2907 011626 005011      CLR    (R1)          ;CLEAR THE FAILED LOCATION
2908 011630 104202 5$:  ERROR +202           ;DUAL ADDRESSING ERROR IN THE UNIBUS MAP
2909 011632 032777 001000 167276 6$: BIT    #BIT09,@SWR   ;SEE IF LOOP ON ERROR IS SET
2910 011640 001366      BNE    20$           ;BRANCH BACK IF SO
2911 011642 077440 7$:  SOB    R4,3$         ;BRANCH IF MORE REGISTERS TO CHECK
2912 011644 005020      CLR    (R0)+         ;CLEAR THE REGISTER JUST TESTED AND POINT TO NEXT REGISTER
2913 011646 005303      DEC    R3            ;DECREMENT LOOP COUNTER AND
2914 011650 001325      BNE    2$            ;BRANCH IF MORE REGISTERS TO TEST
2915 011652 005737 001320      TST    ERRCNT        ;SEE IF THERE WERE ANY ERRORS
```

2916	011656	001405	
2917	011660	005337	001110
2918	011664	005337	001320
2919	011670	104005	

BEQ	RELC22
DEC	\$ERTTL
DEC	ERRCNT
ERROR	+5

:GO TO NEXT SECTION IF NO ERRORS
:DON'T COUNT ERROR +5 AS ANOTHER ERROR
:SAME AS ABOVE
:SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS

```
2920 011672 104414          RELC22: TBITR          ;RESTORE THE T BIT TO ITS CONDITION
2921                          ;BEFORE THE LAST TEST
2922 011674 012700 077406    MOV      #77406,R0      ;MAKE THE KERNEL I-SPACE PAGES ALL
2923                          ;4K, UPWARD EXPANDABLE, READ/WRITE
2924 011700 012701 172300    MOV      #KIPDR0,R1     ;MOVE ADDRESS OF KIPDR0 TO R1
2925 011704 012702 000010    MOV      #8.,R2        ;LOAD 8 PDR'S
2926 011710 010021          1$:  MOV      R0,(R1)+     ;KERNEL I-SPACE PAGE X KIPDRX
2927 011712 077202          SOB      R2,1$         ;SUBTRACT 1 AND BRANCH BACK IF NOT DONE
2928 011714 005000          CLR      R0           ;CLEAR R0
2929 011716 012701 172340    MOV      #KIPAR0,R1     ;MOVE ADDRESS OF KIPAR0 TO R1
2930 011722 012702 000006    MOV      #6,R2         ;LOAD FIRST 6 PAR'S
2931 011726 010021          2$:  MOV      R0,(R1)+     ;MAP KIPARX TO NEXT PHYSICAL PAGE
2932 011730 062700 000200    ADD      #200,R0       ;MAP R0 TO NEXT PAGE
2933 011734 077204          SOB      R2,2$         ;SUBTRACT 1 AND BRANCH BACK IF NOT DONE
2934 011736 012721 170000    MOV      #170000,(R1)+  ;MAP KIPAR6 TO UNIBUS
2935 011742 012721 177600    MOV      #177600,(R1)+  ;MAP KIPAR7 TO I/O PAGE
2936 011746 012737 000001 177572  MOV      #BIT0,MMR0     ;ENABLE FULL 18-BIT MAPPING
2937 011754 012737 000020 172516  MOV      #BIT4,MMR3     ;ENABLE 22-BIT MAPPING
2938                          ;*  AT THIS POINT 22-BIT RELOCATION FROM MEMORY MANAGEMENT
2939                          ;*  IS ENABLED, WITH THE KIPAR'S MAPPED TO PHYSICAL 0-24K.
2940                          ;*  KIPAR6 IS MAPPED TO THE UNIBUS (170000) AND
2941                          ;*  KIPAR7 IS MAPPED TO THE I/O PAGE (177600).
```

2952

.SBTTL TEST # 4 - MAP REGISTER ADDRESS DECODE TEST

*TEST 4 MAP REGISTER ADDRESS DECODE TEST

* THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
* DECODER FOR THE UNIBUS MAP REGISTERS IS STUCK TRUE. KIPAR6 IS
* SET UP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS TO SELECT
* MAPL00, THROUGH KIPAR6. THE TEST THEN CHANGES ONE BIT AT A TIME
* IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPL00. IF IT DOES, AN
* ERROR IS REPORTED.

TST4:

011762	000004				SCOPE	
011764	004737	005166			JSR	PC,PRETST ;GO SET UP PRETEST DATA
011770	012242	012126	000004		.WORD	TST5,20\$,4 ;DATA USED BY PRETST
2953 011776	013737	172354	001200		MOV	KIPAR6,\$TMP2 ;SAVE KIPAR6 FOR RESTORATION LATER
2954 012004	012737	177702	172354		MOV	#177702,KIPAR6 ;PUT MAP REGISTER 0 ADDR IN PAR6
2955 012012	012702	175254			MOV	#175254,R2 ;PATTERN FOR TESTING.
2956 012016	010237	170200			MOV	R2,MAPL0 ;LOAD MAP REGISTER 0
2957 012022	012700	004000			MOV	#BIT11,R0 ;SET BIT 11 TO FLOAT THROUGH PAR6
2958 012026	012704	140000			MOV	#140000,R4 ;VIRT.ADDR. TO SELECT PAR6
2959 012032	074037	172354		1\$:	XOR	R0,KIPAR6 ;CHANGE A BIT OF MAP REGISTER 0'S ADDR
2960 012036	010046				MOV	R0,-(SP) ;SAVE R0 ON STACK
2961 012040	013746	172354			MOV	KIPAR6,-(SP) ;SAVE KIPAR6 ON STACK
2962 012044	012737	000020	001326		MOV	#TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
2963 012052	011401				MOV	(R4),R1 ;READ LOCAT.ON POINTED TO BY PAR6
2964 012054	005037	001326			CLR	CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2965 012060	020201				CMP	R2,R1 ;SEE IF DATA FETCHED MATCHES PATTERN
2966 012062	001047				BNE	4\$;BRANCH IF NOT SAME
2967 012064	012737	000020	001326		MOV	#TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
2968 012072	005014				CLR	(R4) ;TRY TO CLEAR THIS LOCATION
2969 012074	005037	001326			CLR	CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2970 012100	005737	170200			TST	MAPL0 ;SEE IF MAP REGISTER 0 GOT CLEARED
2971 012104	001036				BNE	4\$;BRANCH IF MAP REGISTER NOT ZERO
2972 012106	010237	170200			MOV	R2,MAPL0 ;RESTORE MAPL0
2973 012112	010446				MOV	R4,-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
2974 012114	013746	172354			MOV	KIPAR6,-(SP) ;PUT PAR ON STACK FOR ADREXT SUBROUTINE USE
2975 012120	004737	003674			JSR	PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
2976 012124	000416				BR	2\$;GO CALL ERROR
2977 012126	012737	000020	001326	20\$:	MOV	#TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
2978 012134	011401				MOV	(R4),R1 ;READ LOCATION POINTED TO BY PAR6
2979 012136	005037	001326			CLR	CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2980 012142	020201				CMP	R2,R1 ;SEE IF DATA FETCHED MATCHES PATTERN
2981 012144	001012				BNE	3\$;BRANCH IF NOT SAME
2982 012146	005014				CLR	(R4) ;TRY TO CLEAR THIS LOCATION
2983 012150	005737	170200			TST	MAPL0 ;SEE IF MAP REGISTER 0 GOT CLEARED
2984 012154	001006				BNE	3\$;BRANCH IF MAP REGISTER NOT ZERO
2985 012156	010237	170200			MOV	R2,MAPL0 ;RESTORE MAPL0
2986 012162	104207			2\$:	ERROR	+207 ;GOT TO MAPL0 WITH ONE BIT DIFFERENT IN ADDRESS FROM 1777020
2987 012164	012737	000077	005766		MOV	#77,EADRES+2 ;RESTORE 77 TO EADRES+2
2988 012172	032777	001000	166736	3\$:	BIT	#BIT09,@SWR ;SEE IF LOOP ON ERROR IS SET
2989 012200	001352				BNE	20\$;BRANCH BACK TO LOOP IF SO
2990 012202	012737	000020	001326	4\$:	MOV	#TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
2991 012210	010114				MOV	R1,(R4) ;RELOAD THE LOCATION
2992 012212	005037	001326			CLR	CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
2993 012216	012637	172354		5\$:	MOV	(SP)+,KIPAR6 ;RESTORE KIPAR6

2994	012222	012600				MOV	(SP)+,R0	:RESTORE R0
2995	012224	074037	172354			XOR	R0,KIPAR6	:RESTORE BIT TO ORIGINAL STATUS
2996	012230	006200				ASR	R0	:RIGHT SHIFT ONE PLACE
2997	012232	001277				BNE	1\$:GO CONTINUE TEST IF BIT NOT SHIFTED OUT YET
2998	012234	013737	001200	172354	6\$:	MOV	\$TMP2,KIPAR6	:RESTORE KIPAR6

3011

```
.SBTTL TEST # 5 - DATA PATH, UNIBUS TO MAIN MEMORY
:*****
:*TEST 5 DATA PATH, UNIBUS TO MAIN MEMORY
:*
:* THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA
:* THE UNIBUS. THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO
:* THAT THE ADDRESS IS NOT RELOCATED. THE TEST TRIES TO LOAD THE
:* PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE
:* SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL
:* THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS. IF THIS TEST
:* DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS
:* IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.
:*
:*****
```

```
012242
012242 000004
012244 004737 005166
012250 012630 012262 000005
3012 012256 004737 002772
3013 012262 012704 000035
3014 012266 012737 170400 172354
3015 012274 012737 170400 001210
3016 012302 005037 001330
3017 012306 012737 000020 001326
3018 012314 013700 140000
3019 012320 005037 001326
3020 012324 005737 001330
3021 012330 001411
3022 012332 062737 000200 172354
3023 012340 062737 000200 001210
3024 012346 077423
3025 012350 104010
3026 012352 000574
3027 012354 013737 172354 172352
3028 012362 013737 001210 001206
3029 012370 042737 170000 172352
3030 012376 042737 170000 001206
3031 012404 012737 173214 120000
3032 012412 012737 000020 001326
3033 012420 013701 140000
3034 012424 005037 001326
3035 012430 022701 173214
3036 012434 001406
3037 012436 020001
3038 012440 001004
3039 012442 062737 000200 172354
3040 012450 077464
3041 012452 012701 006102
3042 012456 012702 140000
3043 012462 012703 000007
3044 012466 012737 000020 001326
3045 012474 011112
3046 012476 021112
3047 012500 001426
```

```
TST5:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST6,20$,5 ;DATA USED BY PRETST
JSR PC,CLRMAP ;CLEAR ALL MAP REGISTERS
MOV #35,R4 ;DO 35 ACCESSES
MOV #170400,KIPAR6 ;START WITH ADDRESS 8K FROM UNIBUS
MOV #170400,$TMP6 ;MOVE IT TO $TMP6 ALSO
2$: CLR PCPUER ;CLEAR ERROR CONDITION LOCATION
MOV #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
MOV 140000,R0 ;TRY TO READ ADDRESS POINTED TO BY PAR6
CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
TST PCPUER ;SEE IF READ OF ADDRESS TIMED OUT
BEQ 4$ ;BRANCH IF REFERENCE WAS GOOD
3$: ADD #200,KIPAR6 ;TRY NEXT 4K BLOCK OF MEMORY
ADD #200,$TMP6 ;ADD 200 TO $TMP6 ALSO
SOB R4,2$ ;SUBTRACT 1 FROM R4 AND BRANCH IF NOT DONE
ERROR +10 ;NO UNIBUS ADDRESSES RESPOND
BR SIZEJ0 ;BRANCH TO SIZE JUMPER SECTION
4$: MOV KIPAR6,KIPAR5 ;PUT PAR6 INTO PAR5
MOV $TMP6,$TMP5 ;DO SAME TRANSFER
BIC #170000,KIPAR5 ;MAKE PAR5 A NON UNIBUS ADDRESS
BIC #170000,$TMP5 ;CLEAR SAME BITS
MOV #173214,120000 ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST BUS
MOV #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
MOV 140000,R1 ;READ TEST LOCATION BY UNIBUS
CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
CMP #173214,R1 ;SEE IF DATA WAS READ PROPERLY
BEQ 5$ ;DATA OKAY NOW VERIFY DATA PATH
CMP R0,R1 ;SEE IF DATA CHANGED FROM FIRST READ
BNE 5$ ;BRANCH AROUND NEXT TRY IF SO
5$: ADD #200,KIPAR6 ;TRY NEXT 4K BLOCK OF MEMORY
SOB R4,2$ ;BRANCH BACK TO TRY NEXT 4K BLOCK
MOV #PATRNS,R1 ;LOAD ADDRESS OF BIT PATTERNS IN R1
MOV #140000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
MOV #7,R3 ;DO 7 PATTERNS
MOV #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
6$: MOV (R1),(R2) ;LOAD COUNT INTO TEST LOCATION VIA U.B.
CMP (R1),(R2) ;COMPARE COUNT WITH DATA READ
BEQ 10$ ;BRANCH IF DATA MATCHES
```

3048	012502	011137	001174			MOV	(R1), \$TMP0	:MOVE EXPECTED PATTERN TO \$TMP0
3049	012506	011237	001176			MOV	(R2), \$TMP1	:MOVE RECEIVED PATTERN TO \$TMP1
3050	012512	011246				MOV	(R2), -(SP)	:PUT DATA ON STACK FOR DATEXT SUBROUTINE USE
3051	012514	004737	003624			JSR	PC, DATEXT	:SUBROUTINE TO LOAD DATAOR AND DATAND
3052	012520	011146				MOV	(R1), -(SP)	:PUT PATTERN ON STACK FOR PATEXT SUBROUTINE USE
3053	012522	004737	003650			JSR	PC, PATEXT	:GO SET DATA INTO PATTOR AND PATAND
3054	012526	012737	012536	001106		MOV	#7\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 61\$
3055	012534	000403				BR	8\$:BRANCH OVER LOOP ON ERROR SECTION
3056	012536	011112			7\$:	MOV	(R1), (R2)	:LOAD COUNT INTO TEST LOCATION VIA U.B.
3057	012540	021112				CMP	(R1), (R2)	:COMPARE COUNT WITH DATA READ
3058	012542	001401				BEQ	9\$:BRANCH IF OK NOW
3059	012544	104204			8\$:	ERROR	+204	:REPORT ERROR(S) ON UNIBUS DATA PATH
3060	012546	032777	001000	166362	9\$:	BIT	#BIT9, @SWR	:SEE IF LOOP ON ERROR IS SET
3061	012554	001370				BNE	7\$:BRANCH BACK IF SO
3062	012556	062701	000002		10\$:	ADD	#2, R1	:MOVE TO NEXT PATTERN
3063	012562	077334				SOB	R3, 6\$:DECREMENT LOOP COUNTER AND BRANCH IF NOT DONE
3064	012564	005037	001326			CLR	CPUEXP	:CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3065	012570	005737	001320			TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS TEST
3066	012574	001405				BEQ	11\$:BRANCH IF NO ERRORS ON THIS TEST
3067	012576	005337	001110			DEC	\$ERTTL	:DON'T COUNT ERROR +11 AS ANOTHER ERROR
3068	012602	005337	001320			DEC	ERRCNT	:SAME AS ABOVE
3069	012606	104011				ERROR	+11	:SUMMARY OF ERRORS ON THE UNIBUS DATA PATH
3070	012610	023737	001206	172352	11\$:	CMP	\$TMP5, KIPAR5	:MAKE SURE PARS CONTAINS EXPECTED CONTENTS
3071	012616	001404				BEQ	TST6	:BRANCH IF OK
3072	012620	104030				ERROR	+30	:KIPAR5 NOT LOADED PROPERLY - SKIPPING NEXT TEST
3073	012622	013737	001206	172352		MOV	\$TMP5, KIPAR5	:RESET KIPAR5 FOR EXECUTION OF NEXT TEST

3087

```
.SBTTL TEST # 6 - MAP DOESN'T RELOCATE IF NOT ENABLED
:*****
:*TEST 6 MAP DOESN'T RELOCATE IF NOT ENABLED
:*
:* THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BITS
:* OF MMR3 IS NOT SET. THE TEST ASSUMES THAT THE PREVIOUS TEST HAS
:* RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS
:* MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER
:* THAN OR EQUAL TO MAPREG #2. KIPAR5 IS ALSO POINTING TO THE
:* SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS.
:* THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP
:* AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS. SINCE THE
:* MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.
:*
:*****
```

3088	012630	000004				TST6:	SCOPE		
3089	012632	004737	005166				JSR	PC,PRETST	:GO SET UP PRETEST DATA
3090	012636	012744	012710	000006			.WORD	SIZEJ0,20\$,6	:DATA USED BY PRETST
3091	012644	052737	000001	177572			BIS	#BIT0,MMR0	:TURN MEMORY MANAGEMENT BACK ON
3092	012652	013700	172354				MOV	KIPAR6,R0	:PUT UNIBUS ADDRESS OF MAP REGISTER IN R0
3093	012656	072027	177773				ASH	#-5,R0	:RIGHT SHIFT R0 5 PLACES
3094	012662	042700	177400				BIC	#177400,R0	:CLEAR UPPER BYTE
3095	012666	042737	177000	172352			BIC	#177000,KIPAR5	:MAKE KIPAR5 ACCESS THE FAST BUS
3096	012674	012720	021042				MOV	#021042,(R0)+	:SET BOTTOM BIT IN EACH ALU
3097	012700	012710	000042				MOV	#42,(R0)	:SET BOTTOM BIT IN EACH ALU
3098	012704	005037	120000				CLR	120000	:CLEAR TEST LOCATION VIA FAST BUS
3099	012710	012737	000020	001326	20\$:		MOV	#TIMOUT,CPUEXP	:TIMEOUTS MIGHT OCCUR IN THIS TEST.
3100	012716	012737	043207	140000			MOV	#43207,140000	:LOAD TEST LOCATION VIA UNIBUS THIS LOAD SHOULD NOT BE
3101	012724	005037	001326				CLR	CPUEXP	:RELOCATED BY THE UNIBUS MAP, SINCE BIT05 OF MMR3 IS CLEAR.
3102	012730	013703	120000				MOV	120000,R3	:CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3103	012734	022703	043207				MOV	#43207,R3	:READ TEST LOCATION VIA FAST BUS
3104	012740	001401					CMP	SIZEJ0	:SEE IF DATA MATCHES
3105	012742	104012					BEQ	SIZEJ0	:BRANCH IF DATA GOOD
							ERROR	+12	:MAP RELOCATED WHEN NOT ENABLED

```

3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118 012744 012705 020226
3119 012750 012700 170200
3120 012754 012701 000040
3121 012760 012702 006002
3122 012764 012720 020000
3123 012770 005020
3124 012772 005022
3125 012774 077105
3126 012776 012703 006002
3127 013002 052737 000040 172516
3128 013010 052737 000001 177572
3129 013016 012700 117776
3130 013022 012737 170000 172350
3131 013030 012702 125252
3132 013034 012737 177777 001174
3133 013042 012737 177777 003532
3134 013050 012737 037776 001310
3135 013056 005037 001312
3136 013062 012737 013070 001106
3137 013070 012737 000020 001326
3138 013076 004737 003474
3139 013102 000417
3140 013104 005037 001326
3141 013110 004737 005230
3142 013114 104214
3143 013116 062737 000200 172350
3144 013124 022737 177400 172350
3145 013132 001356
3146 013134 104013
3147 013136 000137 010000
3148 013142 005037 001326
3149 013146 004737 005266
3150 013152 104213
  
```

.SBTTL DETERMINATION OF FIRST USEABLE MAP REGISTER

```

*****
*
* THIS TEST DETERMINES THE SETTING OF THE JUMPERS ON THE UNIBUS
* MAP WHICH ALLOW THE MAP TO RESPOND TO THOSE ADDRESSES BETWEEN
* THE JUMPER RANGE. THE DEFAULT SETTING ALLOWS THE MAP TO RESPOND
* TO ADDRESSES 000000 - 757776 ON THE UNIBUS. IF THE JUMPERS ARE
* NOT SET IN THEIR DEFAULT POSITION AN INFORMATIONAL MESSAGE IS GIVEN.
* >>>>>>>>NOTE<<<<<<<<
* THIS IS THE FIRST TEST IN WHICH THE UNIBUS MAP IS TURNED ON.
*
*****
SIZEJO: MOV #SDDW0,R5 ;PUT ADDRESS OF SDDW0 IN R5
MOV #MAPLO,R0 ;LOAD ADDRESS OF FIRST MAP REGISTER IN R0
MOV #40,R1 ;DO ALL 40 REGISTERS
MOV #SPECST,R2 ;SET R2 TO BEGINING OF SPECIAL STACK
1$: MOV #20000,(R0)+ ;LOAD 4K INTO LOWER 16 BITS AND
CLR (R0)+ ;CLEAR THE UPPER 6 BITS
CLR (R2)+ ;CLEAR THE SPECIAL STACK LOCATION
SOB R1,1$ ;BRANCH IF THERE ARE MORE TO LOAD
MOV #SPECST,R3 ;RESET SPECIAL STACK POINTER
BIS #BIT5,MMR3 ;TURN ON MAP RELOCATION
BIS #BIT0,MMR0 ;MAKE SURE MEMORY MANAGEMENT IS ON
MOV #117776,R0 ;THIS WILL BE USED TO SELECT PAR 4, ADDRESS 17776
MOV #170000,KIPAR4 ;LOAD MAP REGISTER 0 -200 IN KIPAR4
MOV #125252,R2 ;CONSTANT TO LOAD INTO LOCATION 17776
MOV #-1,$TMPO ;MOVE -1 TO MAP REGISTER POINTER
MOV #-1,FTTHRU ;INITIALIZE ONE TIME ENTRANCE FLAG IN SUBROUTINE
MOV #37776,UBM24L ;MOVE TEST CELL ADDRESS TO LOCATION UBM24L
CLR UBM24U ;CLEAR UPPER LOCATION UBM24U
MOV #2$,SLPERR ;MOVE LOOP ON ERROR ADDRESS TO $LPERR
2$: MOV #TIMOUT,CPUEXP ;EXPECTING CPU TIME OUT ON UNIBUS
JSR PC,TSTLOC ;GO TO SUBROUTINE TO SEE IF LOCATION RESPONDS
BR 3$ ;RETURN IS HERE IF LOWEST FOUND
CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
JSR PC,DSABLD ;GO SEE IF REGISTER SHOULD BE DISABLED
ERROR +214 ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
ADD #200,KIPAR4 ;MAP TO NEXT REGISTER
CMP #177400,KIPAR4 ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
BNE 2$ ;GO TEST NEXT MAP REGISTER IF NOT
ERROR +13 ;FATAL ERROR, RESTARTING PROGRAM
JMP START ;JUMP TO RESTART PROGRAM
3$: CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
JSR PC,ENABLD ;GO CHECK TO SEE IF REGISTER SHOULD BE ENABLED
ERROR +213 ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
  
```

```

3151 .SBTTL SECTION TO DETERMINE NUMBER OF USEABLE MAP REGISTERS
3152 :*****
3153 :* THIS SECTION DETERMINES THE NUMBER OF USEABLE MAP REGISTERS BY ACCOM-
3154 :* PLISHING THE SAME LOCATION ACCESS TEST AS IN THE PREVIOUS SECTION.
3155 :*
3156 :*****
3157 013154 013737 001174 001266 SIZEJ1: MOV $TMP0,MMRLOW ;MOVE REGISTER NUMBER FOUND USEABLE TO MMRLOW
3158 013162 013737 172350 001256 MOV KIPAR4,LOWEST ;MOVE LOWEST USEABLE REGISTER TO LOWEST
3159 013170 022737 170000 001256 CMP #170000,LOWEST ;SEE IF LOWEST REGISTER FOUND WAS THE LOWEST
3160 013176 001427 BEQ 1$ ;BRANCH AROUND SETUP IF IT WAS
3161 013200 005037 001272 CLR UBRLOW ;MAP REGISTER 0 IS LOWEST FOR UNIBUS MEMORY
3162 013204 012737 170000 001262 MOV #170000,UBMLOW ;MOVE PAR VALUE OF UB MEMORY TO UBMLow
3163 013212 013737 001174 001274 MOV $TMP0,UBRHI ;MOVE REGISTER NUMBER FOUND USEABLE TO UBRHI
3164 013220 005337 001274 DEC UBRHI ;POINT IT AT HIGHEST UB MEMORY MAP REGISTER
3165 013224 013737 172350 001264 MOV KIPAR4,UBMHI ;MOVE KIPAR4 TO UNIBUS MAP HIGHEST AND
3166 013232 162737 000200 001264 SUB #200,UBMHI ;SUBTRACT 200 FROM IT TO POINT TO LAST USEABLE UBMEM PAGE
3167 013240 012737 177400 001260 MOV #177400,HIGEST ;MOVE HIGHEST REGISTER TO HIGEST AND
3168 013246 012737 000031 001270 MOV #31,MMRHI ;POINT TO LAST USEABLE MAP REGISTER
3169 013254 000532 BR YESMSG ;GO TYPE MESSAGE OF NON-DEFAULT INFORMATION
3170 013256 012737 013324 001106 1$: MOV #3$, $LPERR ;SET LOOP ON ERROR TO 3$
3171 013264 062737 000200 172350 ADD #200,KIPAR4 ;MAP TO NEXT REGISTER
3172 013272 000414 BR 3$ ;BRANCH OVER POST-TSTLOC CODE
3173 013274 005037 001326 2$: CLR CPUEXP ;CLEAR CPUEXP - DON'T EXPECT TRAPS FOR A WHILE
3174 013300 062737 000200 172350 ADD #200,KIPAR4 ;MAP TO NEXT REGISTER
3175 013306 022737 177400 172350 CMP #177400,KIPAR4 ;SEE IF ALL MAP REGISTERS HAVE BEEN TRIED
3176 013314 001416 BEQ 4$ ;BRANCH IF ALL ARE DONE
3177 013316 004737 C05266 JSR PC,ENABLD ;GO SEE IF MAP REGISTER SHOULD BE ENABLED
3178 013322 104213 ERROR +213 ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
3179 013324 012737 000020 001326 3$: MOV #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3180 013332 004737 003474 JSR PC,TSTLOC ;GO TO SUBROUTINE TO SEE IF IT RESPONDS
3181 013336 000756 BR 2$ ;RETURN IS HERE IF IT WAS LOADED
3182 013340 005037 001326 CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3183 013344 004737 005230 JSR PC,DSABLD ;RETURN IS HERE IF NOT LOADED - GO SEE IF
3184 ;REGISTER SHOULD BE DISABLED
3185 013350 104214 ERROR +214 ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
3186 013352 013737 001174 001270 4$: MOV $TMP0,MMRHI ;MOVE REGISTER FOUND TO MMRHI
3187 013360 005337 001270 DEC MMRHI ;DECREMENT THIS VALUE - IT IS ONE TOO MANY
3188 013364 013737 172350 001260 MOV KIPAR4,HIGEST ;MOVE FIRST UNUSABLE REGISTER TO HIGEST
3189 013372 023727 001260 177400 CMP HIGEST,#177400 ;SEE IF UPPER JUMPER IS DEFAULT.
3190 013400 001522 BEQ NOMSG ;BRANCH AROUND MESSAGE TYPEOUT IF IT IS DEFAULT

```

```

3191 .SBTTL DETERMINE UNUSEABLE MAP REGISTER WINDOW SIZE
3192 :*****
3193 :* THIS SECTION PERFORMS THE SAME LOCATION ACCESS TEST PREVIOUSLY DONE
3194 :* LOOKING FOR THE FIRST ACCESSIBLE REGISTER, OR THE UPPER LIMIT. ONCE
3195 :* FOUND, THE SIZE OF THE WINDOW HAS BEEN DETERMINED AND THE MESSAGE
3196 :* TELLING THE USER OF THE SIZE IS PRINTED.
3197 :*
3198 :*****
3199 013402 013737 001260 001262 SIZEJ2: MOV HIGEST,UBMLOW ;MOVE UPPER LIMIT TO UBMLOW LOCATION
3200 013410 062737 000200 001262 ADD #200,UBMLOW ;POINT UBMLOW TO FIRST USABLE UNIBUS MEM PAGE
3201 013416 013737 001174 001272 MOV $TMP0,UBRLOW ;MOVE REGISTER NUMBER TO UBRLOW
3202 013424 012737 013460 001106 MOV #2$, $LPERR ;SET LOOP ON ERROR TO 2$
3203 013432 000412 BR 2$ ;BRANCH OVER CHECK FOR NON-EXISTENT LOCATION
3204 013434 062737 000200 172350 1$: ADD #200,KIPAR4 ;MAP TO NEXT REGISTER
3205 013442 022737 177400 172350 CMP #177400,KIPAR4 ;SEE IF WE ARE POINTING JUST BELOW THE I/O PAGE
3206 013450 001414 BEQ 3$ ;GO INITIALIZE UBMHI IF WE ARE
3207 013452 004737 005230 JSR PC,DSABLD ;GO SEE IF LOCATION SHOULD BE DISABLED
3208 013456 104214 ERROR +214 ;MAP REGISTER DISABLED WHEN SHOULD BE ENABLED
3209 013460 012737 000020 001326 2$: MOV #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
3210 013466 004737 003474 JSR PC,TSTLOC ;GO TO SUBROUTINE TO SEE IF IT DOESN'T RESPOND
3211 013472 000403 BR 3$ ;RETURN IS HERE IF IT WAS LOADED - GO INITIALIZE UBMHI
3212 013474 005037 001326 CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3213 013500 000755 BR 1$ ;RETURN IS HERE IF NOT - GO BACK FOR ANOTHER TRY
3214 ;AT THIS POINT, KIPAR4 POINTS JUST ABOVE HIGHEST ADDRESS OF UNIBUS MEMORY
3215 013502 005037 001326 3$: CLR CPUEXP ;CLEAR CPUEXP - NO TIMEOUTS FOR A WHILE
3216 013506 004737 005266 JSR PC,ENABLD ;GO SEE IF MAP REGISTER SHOULD BE ENABLED
3217 013512 104213 ERROR +213 ;MAP REGISTER ENABLED WHEN SHOULD BE DISABLED
3218 013514 013737 172350 001276 MOV KIPAR4,BUPWIN ;MOVE THIS VALUE TO 'B'EGINING 'UP'PER 'WIN'DOW
3219 013522 013737 172350 001264 MOV KIPAR4,UBMHI ;MOVE THIS TO UBMHI ALSO
3220 013530 013737 001174 001274 MOV $TMP0,UBRHI ;MOVE MAP REGISTER POINTER TO UBRHI
3221 013536 005337 001274 DEC UBRHI ;DECREMENT THIS VALUE - IT IS ONE TOO MANY
  
```

```

3222          .SBTTL  ROUTINE TO PRINT THE WINDOW SIZE MESSAGE
3223          :*****
YESMSG: TST     $PASS          ;SEE IF THIS IS FIRST PASS
        BNE     NOMSG         ;BRANCH TO NEXT SECTION IF NOT FIRST PASS
        TYPE    ,JMPMSG       ;TYPE SIZE JUMPERS NOT IN DEFAULT - FOR INFO ONLY
        MOV     #DTMS,R0      ;SET UP MESSAGE POINTER
        MOV     #1$,-(SP)     ;PUSH RETURN ON THE STACK
        MOV     R0,-(SP)      ;PUSH R0 ON THE STACK
        MOV     R1,-(SP)      ;PUSH R1 ON THE STACK
        JMP     TYPDAT        ;GO TYPE THE DATA
1$:      TYPE    , $CRLF       ;TYPE A <CRLF>
        TYPE    , $CRLF       ;TYPE ONE MORE <CRLF>
        CMP     #SPECST,R3    ;SEE IF ANY TIMEOUTS OCCURED
        BEQ     NOMSG         ;BRANCH TO NEXT SECTION IF NONE
        TYPE    ,TOMSG        ;TYPE THE TIMEOUTS MESSAGE
        MOV     #SPECST,R3    ;RESET R3
2$:      MOV     -(R3),-(SP)   ;PUSH THE REGISTER # ONTO THE STACK IN ORDER IT WAS PUSHED
3$:      TYPDS          ;TYPE THE NUMBER IN DECIMAL, LEADING ZEROS SUPPRESSED
        TYPE    , $CRLF       ;TYPE A <CRLF>
        TST     -(R3)         ;SEE IF THERE IS ANOTHER REGISTER NUMBER TO PRINT
        BEQ     4$            ;BRANCH AROUND SETUP IF NONE
        MOV     (R3),-(SP)    ;PUSH THIS NUMBER ON THE STACK
        BR      3$           ;BRANCH BACK TO PRINT IT
4$:      TYPE    , $CRLF       ;TYPE ONE MORE <CRLF>

```

```

3246
3247
3248
3249
3250
3251
3252
3253 013646 010546
3254 013650 013705 001256
3255 013654 042705 170000
3256 013660 072527 177773
3257 013664 052705 170200
3258 013670 010537 001300
3259 013674 012605
3260 013676 013737 001300 001302
3261 013704 062737 000002 001302

```

```

.SBTTL SETUP POINTERS FOR LOWEST AND HIGHEST MAP REGISTERS
*****
:
:
:
:
:
:
:
:
:
:
:
*****
NOMSG: MOV R5, -(SP) ;SAVE R5
MOV LOWEST,R5 ;MOVE PAR DATA TO R5 FOR CONVERSION
BIC #170000,R5 ;CLEAR BITS 15 TO 12
ASH #-5,R5 ;SHIFT INDICATOR BITS TO THE RIGHT 5 PLACES
BIS #170200,R5 ;FORM ADDRESS
MOV R5,LREGL ;SAVE RESULTS
MOV (SP)+,R5 ;RESTORE R5
MOV LREGL,I REGU ;MOVE ADDRESS TO LREGU
ADD #2,LREGU ;POINT TO UPPER 6 BITS OF MAP REG

```


3280

.SBTTL TEST # 7 - ENSURE THAT THERE IS NO DUAL MAPPING
 :*****
 :*TEST 7 ENSURE THAT THERE IS NO DUAL MAPPING
 :*

THIS TEST VERIFIES THAT THERE IS NO DUAL MAPPING. IT CLEARS ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS THAT ONE WITH 00040000. IF MAP RELOCATION IS ENABLED (AND IN THIS TEST IT IS), SUBROUTINE CLRMAP CLEARS ALL BUT THE LOWER 16 BITS OF MAPL1, AND LOADS 20000 THERE. THIS IS SO THAT APT, IF CONTROLLING THIS DIAGNOSTIC, CAN STILL EXAMINE THE PROPER LOCATIONS. THE TEST THEN USES A VIRTUAL ADDRESS TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD REFERENCE ADDRESS 00057776 (00037776 IF MAPL1 CONTAINS 20000 AS PER CONDITIONS DESCRIBED ABOVE). A REFERENCE IS MADE THROUGH EACH OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE CHECKED TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST. IF NOT, BOTH THE MAP REGISTER UNDER TEST AND THE DUALED REGISTER ARE REPORTED.

:*****
 TST7:

013712	000004				SCOPE		
013712	004737	005166			JSR	PC,PRETST	:GO SET UP PRETEST DATA
013714	014616	014260	000007		.WORD	TST10,20\$,7	:DATA USED BY PRETST
3281 013720	005037	172516			CLR	MMR3	:CLEAR MMR3
3282 013726	005037	001312			CLR	UBM24U	:CLEAR THE UPPER EXPECTED LMA LOCATION
3283 013732	052737	000060	172516		BIS	#60,MMR3	:ENABLE 22-BIT ADDRESSING AND MAP RELOCATION
3284 013736	004737	002772			JSR	PC,CLRMAP	:CLEAR ALL MAP REGISTERS
3285 013744	042737	000001	177572		BIC	#1,MMR0	:TURN OFF MEMORY MANAGEMENT
3286 013750	005037	001174			CLR	\$TMP0	:\$TMP0 IS USED AS A FLAG IN THIS TEST
3287 013756	012703	117776			MOV	#117776,R3	:SELECT P.A.R. 4 OFFSET OF 17776
3288 013762	013702	001300			MOV	LREGL,R2	:PUT ADDRESS OF LOWEST USABLE MAP REGISTER IN R2
3289 013766	013700	001256			MOV	LOWEST,R0	:LOAD PAR POINTING TO MAP REGISTER UNDER TEST IN R0
3290 013772	052737	000001	177572		BIS	#1,MMR0	:MAKE SURE MEMORY MANAGEMENT IS ON
3291 014004	005037	001320			CLR	ERRCNT	:CLEAR THE ERROR COUNT FOR ERROR 202 BELOW
3292 014004	013737	001256	172350	100\$:	MOV	LOWEST,KIPAR4	:PAR OF LOWEST USEABLE MAP REGISTER IS LOADED IN KIPAR4
3293 014016	022702	170204			CMP	#MAPL01,R2	:SEE IF WE ARE POINTING AT MAPL1
3294 014022	001003				BNE	1\$:BRANCH IF NOT
3295 014024	022712	020000			CMP	#20000,(R2)	:SEE IF IT CONTAINS 20000 (FOR APT USE)
3296 014030	001410				BEQ	2\$:BRANCH IF SO
3297 014032	012712	040000		1\$:	MOV	#40000,(R2)	:LOAD MAP REGISTER UNDER TEST WITH 8K BASE
3298 014036	010237	057776			MOV	R2,57776	:LOAD TEST LOCATION WITH THE ADDRESS
3299							:OF THE MAP REGISTER UNDER TEST
3300 014042	012737	057776	001314		MOV	#57776,UBM24P	:MOVE ANTICIPATED ADDRESS TO LOCATION UBM24P
3301 014050	000405				BR	3\$:BRANCH OVER LOCATION SETUP
3302 014052	010237	037776		2\$:	MOV	R2,37776	:LOAD TEST LOCATION WITH THE ADDRESS
3303							:OF THE MAP REGISTER UNDER TEST
3304 014056	012737	037776	001314		MOV	#37776,UBM24P	:MOVE ANTICIPATED ADDRESS TO LOCATION
3305 014064	162737	000200	172350	3\$:	SUB	#200,KIPAR4	:PREPARE KIPAR4 FOR FIRST 'ADD 200'
3306 014072	062737	000200	172350	4\$:	ADD	#200,KIPAR4	:TRY NEXT MAP REGISTER
3307 014100	005037	001330		41\$:	CLR	PCPUER	:CLEAR THE ERROR RECEIVER
3308 014104	012737	017776	001310		MOV	#17776,UBM24L	:MOVE DEFAULT LMA ADDRESS TO UBM24L
3309 014112	020037	172350			CMP	R0,KIPAR4	:SEE IF REGISTER UNDER TEST IS POINTED TO BY KIPAR4
3310 014116	001003				BNE	45\$:BRANCH AROUND SETUP IF NOT
3311 014120	013737	001314	001310		MOV	UBM24P,UBM24L	:MOVE SPECIAL LMA ADDRESS TO LOCATION
3312 014126	012737	000020	001326	45\$:	MOV	#TIMOUT,CPUEXP	:POSSIBLE NON-EXISTENT MEMORY
3313 014134	011304				MOV	(R3),R4	:READ THROUGH THE MAP REGISTER

3314	014136	005037	001326		CLR	CPUEXP	:NO MORE TIMEOUTS FOR AWHILE	
3315	014142	005037	001320		CLR	ERRCNT	:CLEAR ERROR COUNT	
3316	014146	005737	001330		TST	PCPUER	:SEE IF THERE WAS AN ERROR	
3317	014152	001110			BNE	8\$:BRANCH AROUND DATA TEST IF SO	
3318	014154	020402			CMP	R4,R2	:SEE IF CORRECT DATA WAS FETCHED	
3319	014156	001075			BNE	6\$:BRANCH IF NO MATCH	
3320	014160	032777	004000	164750	BIT	#BIT11,@SWR	:SEE IF THIS IS AN 11/24 WITH UB MEM ONLY	
3321	014166	001406			BEQ	49\$:BRANCH IF NOT	
3322	014170	013737	177736	001304	MOV	LMAHI,LMAH	:SAVE LMA HIGH REG IN LMAH	
3323	014176	013737	177734	001306	MOV	LMALOW,LMAL	:SAVE LMA LOW REG IN LMAL	
3324	014204	020037	172350		49\$:	CMP	R0,KIPAR4	:SEE IF MAP REGISTERS ARE THE SAME
3325	014210	001455			BEQ	5\$:BRANCH IF CORRECT MAP REGISTER WAS USED	
3326	014212	004737	005076		JSR	PC,CHKLMA	:GO CHECK FOR 11/24 WITH UB MEMORY ONLY	
3327	014216	001304	001306		.WORD	LMAH,LMAL	:ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS	
3328	014222	000450			BR	5\$:RETURN IS HERE IF MATCH ACHIEVED	
3329	014224	011337	001202		MOV	(R3),\$TMP3	:SAVE CONTENTS FOR ERROR PRINTING	
3330	014230	013746	001202		MOV	\$TMP3,-(SP)	:MOVE FAULTY DATA TO STACK FOR SUBROUTINE USE	
3331	014234	013746	172350		MOV	KIPAR4,-(SP)	:MOVE PAR4 CONTENTS TO STACK FOR SUBROUTINE USE	
3332	014240	004737	003674		JSR	PC,ADREXT	:GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND	
3333	014244	010246			MOV	R2,-(SP)	:PUT ADDRESS OF REGISTER ON STACK FOR DATEXT USE	
3334	014246	013746	172356		MOV	KIPAR7,-(SP)	:MOVE PAR CONTENTS TO STACK FOR SUBROUTINE USE	
3335	014252	004737	003624		JSR	PC,DATEXT	:GO SET DATA IN DATAOR AND DATAND	
3336	014256	000424			BR	46\$:BRANCH OVER LOOP ON ERROR SETUP	
3337	014260	020402			20\$:	CMP	R4,R2	:SEE IF CORRECT DATA WAS FETCHED
3338	014262	001023			BNE	47\$:BRANCH IF NO MATCH	
3339	014264	032777	004000	164644	BIT	#BIT11,@SWR	:SEE IF THIS IS AN 11/24 WITH UB MEM ONLY	
3340	014272	001406			BEQ	59\$:BRANCH IF NOT	
3341	014274	013737	177736	001304	MOV	LMAHI,LMAH	:SAVE LMA HIGH REG IN LMAH	
3342	014302	013737	177734	001306	MOV	LMALOW,LMAL	:SAVE LMA LOW REG IN LMAL	
3343	014310	020037	172350		59\$:	CMP	R0,KIPAR4	:SEE IF MAP REGISTERS ARE THE SAME
3344	014314	001406			BEQ	47\$:BRANCH IF CORRECT MAP REGISTER WAS USED	
3345	014316	004737	005076		JSR	PC,CHKLMA	:GO CHECK FOR 11/24 WITH UB MEMORY ONLY	
3346	014322	001304	001306		.WORD	LMAH,LMAL	:ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS	
3347	014326	000401			BR	47\$:RETURN IS HERE IF MATCH ACHIEVED	
3348	014330	104210			46\$:	ERROR	+210	:DUAL MAPPING ERROR IN THE UNIBUS MAP
3349	014332	032777	001000	164576	47\$:	BIT	#BIT9,@SWR	:SEE IF LOOP ON ERROR IS SET
3350	014340	001347			BNE	20\$:BRANCH BACK IF SO	
3351	014342	000414			BR	8\$:BRANCH AROUND ADDRESS MATCH SETTING	
3352	014344	005237	001174		5\$:	INC	\$TMP0	:SET FLAG WHEN ADDRESSES MATCH
3353	014350	000411			BR	8\$:BRANCH AROUND SUBROUTINE CHECK - NOT NEEDED	
3354	014352	004737	005076		6\$:	JSR	PC,CHKLMA	:GO CHECK FOR 11/24 WITH UB MEMORY ONLY
3355	014356	001304	001306		.WORD	LMAH,LMAL	:ADDRESSES OF LOCATIONS CONTAINING LMA CONTENTS	
3356	014362	000401			BR	7\$:RETURN IS HERE IF MATCH ACHIEVED	
3357	014364	000403			BR	8\$:BRANCH OVER LMA SPECIFIC COMPARE CODE	
3358	014366	020037	172350		7\$:	CMP	R0,KIPAR4	:SEE IF MAP REGISTERS ARE THE SAME
3359	014372	001764			BEQ	5\$:BRANCH IF CORRECT MAP REGISTER WAS USED	
3360	014374	022737	177400	172350	8\$:	CMP	#177400,KIPAR4	:SEE IF LAST REGISTER HAS BEEN TRIED
3361	014402	001420			BEQ	10\$:BRANCH TO CONTINUE IF SO	
3362	014404	023737	001260	172350		CMP	HIGEST,KIPAR4	:SEE IF ALL HAVE BEEN TRIED
3363	014412	001227			BNE	4\$:BRANCH IF STILL MORE TO TRY	
3364	014414	062737	000200	172350	9\$:	ADD	#200,KIPAR4	:MAP TO NEXT MAP REGISTER
3365	014422	022737	177400	172350		CMP	#177400,KIPAR4	:SEE IF WE ARE AT THE TOP
3366	014430	001405			BEQ	10\$:BRANCH TO CONTINUE IF SO	
3367	014432	023737	001276	172350		CMP	BUPWIN,KIPAR4	:SEE IF WE ARE POINTING TO THE UPPER WINDOW START
3368	014440	001365			BNE	9\$:BRANCH BACK FOR ANOTHER INCREMENTING SET IF NOT	
3369	014442	000616			BR	41\$:GO BACK FOR A NEW TRY	
3370	014444	005737	001174		10\$:	TST	\$TMP0	:SEE THAT THERE WAS A SUCCESSFUL MATCH

3371	014450	001006		BNE	11\$:BRANCH IF THERE WAS
3372	014452	010246		MOV	R2,-(SP)	:PUT ADDRESS OF REGISTER ON STACK FOR ADREXT USE
3373	014454	013746	172356	MOV	KIPAR7,-(SP)	:PUT PAR ON STACK FOR SUBROUTINE ADREXT USE
3374	014460	004737	003674	JSR	PC,ADREXT	:GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
3375	014464	104210		ERROR	+210	:DUAL MAPPING ERROR IN THE UNIBUS MAP
3376	014466	005037	001174	11\$: CLR	\$TMP0	:CLEAR FLAG FOR NEXT REGISTER
3377	014472	022702	170204	CMP	#MAPL01,R2	:SEE IF R2 IS POINTING TO MAPL1
3378	014476	001006		BNE	12\$:BRANCH IF NOT
3379	014500	022712	040000	CMP	#40000,(R2)	:DOES MAPL1 CONTAIN 40000?
3380	014504	001403		BEQ	12\$:BRANCH IF IT DOES
3381	014506	005037	037776	CLR	37776	:CLEAR LOCATION 37776 ONLY - MAPL1 IS TO BE LEFT ALONE
3382	014512	000401		BR	13\$:SKIP OVER REGISTER CLEAR STEP
3383	014514	005012		12\$: CLR	(R2)	:CLEAR MAP REGISTER JUST TESTED
3384	014516	062700	000200	13\$: ADD	#200,R0	:POINT TO NEXT MAP REGISTER UNDER TEST
3385	014522	062702	000004	ADD	#4,R2	:POINT TO NEXT MAP REGISTER TO LOAD
3386	014526	022700	177400	CMP	#177400,R0	:SEE IF LAST REGISTER HAS BEEN TRIED
3387	014532	001421		BEQ	15\$:BRANCH TO NEXT SECTION IF SO
3388	014534	023700	001260	CMP	HIGEST,R0	:SEE IF ALL MAP REGS HAVE BEEN TESTED
3389	014540	001402		BEQ	14\$:BRANCH IF NO MORE TO TEST
3390	014542	000137	014010	JMP	100\$:JUMP TO BEGIN AGAIN
3391	014546	062700	000200	14\$: ADD	#200,R0	:POINT TO NEXT MAP REGISTER UNDER TEST
3392	014552	062702	000004	ADD	#4,R2	:POINT TO NEXT MAP REGISTER TO LOAD
3393	014556	022700	177400	CMP	#177400,R0	:SEE IF LAST REGISTER HAS BEEN TRIED
3394	014562	001405		BEQ	15\$:BRANCH TO NEXT SECTION IF SO
3395	014564	020037	001276	CMP	R0,BUPWIN	:SEE IF WE ARE POINTING TO UPPER WINDOW START
3396	014570	001336		BNE	11\$:BRANCH FOR ANOTHER INCREMENT SET IF NOT
3397	014572	000137	014072	JMP	4\$:JUMP BACK FOR ANOTHER RUN
3398	014576	005737	001320	15\$: TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
3399	014602	001405		BEQ	TST10	:BRANCH TO NEXT TEST IF NO ERRORS
3400	014604	005337	001110	DEC	\$ERTTL	:DON'T COUNT ERROR +16 AS ANOTHER ERROR
3401	014610	005337	001320	DEC	ERRCNT	:SAME AS ABOVE
3402	014614	104016		ERROR	+16	:SUMMARY OF DUAL MAPPING ERRORS

3411

```

.SBTTL TEST # 10 - LOAD LOC'S 40000-77776 WITH THEIR ADRES'S
:*****
:TEST 10 LOAD LOC'S 40000-77776 WITH THEIR ADRES'S
:
: THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
: ADDRESS 000077776 WITH ITS OWN ADDRESS. IT THEN CHECKS THAT
: MEMORY OVER THE UNIBUS AND LOGS AND REPORTS ANY ERRORS THAT
: IT FINDS.
:*****

```

```

014616
014616 000004
014620 004737 005166
014624 015072 015030 000010
3412 014632 042737 000040 172516
3413 014640 012737 000400 172350
3414 014646 012700 040000
3415 014652 012701 100000
3416 014656 012702 010000
3417 014662 010021
3418 014664 062700 000002
3419 014670 077204
3420 014672 062737 000200 172350
3421 014700 022737 001000 172350
3422 014706 101361
3423
3424
3425
3426 014710 022737 171000 172354
3427
3428 014716 101465
3429 014720 013700 172354
3430
3431 014724 072027 000006
3432 014730 012701 140000
3433 014734 012702 010000
3434 014740 011103
3435 014742 020003
3436 014744 001012
3437 014746 022120
3438 014750 077205
3439 014752 062737 000200 172354
3440 014760 022737 171000 172354
3441 014766 101360
3442 014770 000430
3443 014772 011146
3444 014774 013746 172354
3445 015000 004737 003674
3446 015004 010146
3447 015006 013746 172344
3448 015012 004737 003624
3449 015016 010337 005770
3450 015022 005037 005772
3451 015026 000403
3452 015030 011103
3453 015032 020003
3454 015034 001401

```

```

TST10:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST11,20$,10 ;DATA USED BY PRETST
BIC #BIT5,MMR3 ;TURN OFF MAP RELOCATION
MOV #400,KIPAR4 ;MAP PAGE 4 TO 8K
MOV #40000,R0 ;STARTING ADDRESS FOR DATA PATTERN
1$: MOV #100000,R1 ;VIRTUAL ADDRESS
MOV #4096,R2 ;LOAD 4096 LOCATIONS AT A TIME
2$: MOV R0,(R1)+ ;LOAD PHY. ADDR. INTO EACH MEMORY LOC.
ADD #2,R0 ;POINT TO NEXT PHYSICAL ADDRESS
SOB R2,2$ ;BRANCH IF 4K OF MEMORY NOT LOADED
ADD #200,KIPAR4 ;POINT TO NEXT 4K BANK OF MEMORY
CMP #1000,KIPAR4 ;SEE IF 16K IS LOADED
BHI 1$ ;BRANCH IF MORE MEMORY TO LOAD

:
: MEMORY FROM 8K - 16K IS NOW LOADED WITH ITS OWN ADDRESS
:
CMP #171000,KIPAR6 ;DID I USE ANY MAP REGISTER
;BELOW REGISTER 6 (UB. ADDR 100000)
BLOS TST11 ;BRANCH TO NEXT TEST IF NOT
MOV KIPAR6,R0 ;LOAD PAR6 INTO R0 TO GET
;THE STARTING DATA PATTERN
3$: ASH #6,R0 ;R0 NOW HOLDS THE STARTING DATA PATTERN
MOV #140000,R1 ;STARTING VIRTUAL ADDRESS
MOV #4096,R2 ;PREPARE TO READ 4K AT A TIME
4$: MOV (R1),R3 ;READ MAIN MEMORY THROUGH UNIBUS
CMP R0,R3 ;SEE IF THE ADDRESSES MATCH
BNE 6$ ;BRANCH IF ERROR
5$: CMP (R1)+,(R0)+ ;CHANGE VIRTUAL & PHYSICAL ADDRESSES
SOB R2,4$ ;BRANCH IF 4K OF MEMORY NOT READ
ADD #200,KIPAR6 ;POINT TO NEXT BANK OF 4K THROUGH UNIBUS
CMP #171000,KIPAR6 ;SEE IF THIS POINTS TO 16K PLUS 2
BHI 3$ ;BRANCH IF 16K OF MEMORY NOT CHECKED
BR 10$ ;TEST FINISHED, BRANCH TO EXIT
6$: MOV (R1),-(SP) ;PUT VIRTUAL ADDRESS ON STACK FOR ADREXT SUBROUTINE USE
MOV KIPAR6,-(SP) ;PUT PAR6 CONTENTS ON STACK FOR SUBROUTINE USE
JSR PC,ADREXT ;GO SET DATA IN THE 4 WORDS OF ADDROR AND ADRAND
MOV R1,-(SP) ;PUT DATA ON STACK FOR DATEXT SUBROUTINE
MOV KIPAR2,-(SP) ;MOVE PAR CONTENTS TO STACK FOR SUBROUTINE USE
JSR PC,DATEXT ;SUBROUTINE TO LOAD DATAOR AND DATAND
MOV R3,EADRS2 ;MOVE DATA IN R3 TO EADRS2 FOR ERROR CALL
CLR EADRS2+2 ;CLEAR UPPER 6 BITS TO PRINT
BR 7$ ;BRANCH OVER LOOP ON ERROR SETUP
20$: MOV (R1),R3 ;READ MAIN MEMORY THROUGH UNIBUS
CMP R0,R3 ;SEE IF THE ADDRESSES MATCH
BEQ 8$ ;BRANCH IF OK

```

3455	015036	104205		7\$:	ERROR	+205		:DIDN'T READ ADDRESSES CORRECTLY FROM UNIBUS
3456	015040	032777	001000	164070	8\$:	BIT	#BIT9,@SWR	:SEE IF LOOP ON ERROR IS SET
3457	015046	001370				BNE	20\$:BRANCH BACK IF SO
3458	015050	000736				BR	5\$:CONTINUE TESTING
3459	015052	005737	001320		10\$:	TST	ERRCNT	:WERE THERE ANY ERRORS ON THIS TEST?
3460	015056	001405				BEQ	TST11	:BRANCH IF NO ERRORS ON THIS TEST
3461	015060	005337	001110			DEC	\$ERTTL	:DON'T COUNT ERROR +14 AS ANOTHER ERROR
3462	015064	005337	001320			DEC	ERRCNT	:SAME AS ABOVE
3463	015070	104014				ERROR	+14	:SUMMARY OF UNIBUS ADDRESS FAILURES

3472

```
.SBTTL TEST # 11 - MAIN MEMORY TIMEOUT THROUGH MAP
:*****
:*TEST 11      MAIN MEMORY TIMEOUT THROUGH MAP
:*
:*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
:*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY.  IT USES THE LOWEST
:*      USABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
:*      ZERO.
:*****
```

```
015072
015072 000004
015074 004737 005166
015100 015140 015132 000011
3473 015106 052737 000040 172516
3474 015114 005037 004312
3475 015120 012737 000074 001312
3476 015126 005037 001310
3477 015132 004737 020014
3478 015136 104015
```

```
TST11:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST12,20$,11 ;DATA USED BY PRETST
BIS #BIT5,MMR3 ;TURN MAP RELOCATION BACK ON
CLR LOEFLG ;CLEAR LOEFLG FOR ERROR LOOP INDICATOR
MOV #74,UBM24U ;EXPECTING 74 IN UPPER POSITION OF 11/24 LMA
CLR UBM24L ;EXPECTING ZERO IN LOWER POSITION
20$: JSR PC,MMTOTM ;GO DO THE TEST
ERROR +15 ;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT
```

3489

```
.SBTTL TEST # 12 - RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
:*****
:*TEST 12 RELOC USING LOWEST USABLE MAP REG THRU UNIBUS
:*
:* THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
:* MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
:* IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
:* THIS TEST WILL USE THE LOWEST USABLE MAP REGISTER.
:* IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
:* THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
:*
:*****
```

```
015140
015140 000004
015142 004737 005166
015146 015174 015154 000012
3490 015154 005037 004312
3491 015160 005037 001312
3492 015164 004737 004106
3493 015170 104211
3494 015172 000774
```

```
TST12:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST13,20$,12 ;DATA USED BY PRETST
CLR LOEFLG ;CLEAR LOEFLG - FLAG USED FOR ERROR RETURN
CLR UBM24U ;CLEAR UPPER LOCATION
1$: JSR PC,MAPADD ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE FOR ERROR
BR 1$ ;GO BACK TO THE SUBROUTINE
```

3503

.SBTTL TEST # 13 - CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
:*****
:*TEST 13 CARRY PROP OF MAP'S RELOC ADDER THRU UNIBUS
:*

* EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
* WITH 00030000 UP TO 17000000. THAT IS, THE FIRST OF EVERY 2K
* WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
* WORKING PROPERLY .
*
:*****

3504 015174 000004
3504 015176 004737 005166
3505
3506
3507
3508
3509 015202 015236 015210 000013
3510 000040
3511 015210 005037 004312
3512 015214 005037 001312
3513 015220 012737 020000 001310
3514 015226 004737 004416
3515 015232 104211
3516 015234 000774
3517 015236

TST13: SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
:*****
:**IMPORTANT** : IF THE POSITION OF THIS TEST IS CHANGED, CHANGE THE THIRD
: WORD BELOW TO THE NEW TEST NUMBER THIS TEST WILL OCCUPY.
:*****
: .WORD 2\$,20\$,13 ;DATA USED BY PRETST
NEXMEM=BIT5 ;BIT05 IS NON-EXISTENT MEMORY BIT IN THE CPU ERROR REGISTER
20\$: CLR LOEFLG ;CLEAR LOEFLG - USED AS AN ERROR RETURN FLAG
CLR UBM24U ;CLEAR UPPER LOCATION
MOV #20000,UBM24L ;PUT 20000 IN LOWER LOCATION
1\$: JSR PC,TCPMRA ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE IF AN ERROR
BR 1\$;RETURN TO THE TEST
2\$:


```

3518                                     .SBTTL UNIBUS MAP SETUP
3519                                     :*****
3520                                     : THE NEXT 3 TESTS ARE RUN THROUGH THE UNIBUS MAP
3521                                     :*****
3522
3523 015236 000004 UBMSU: SCOPE ;LOOP ON PREVIOUS TEST
3524 015240 105337 001100 DECB $STSNM ;DECREMENT $STSNM - THIS IS NOT A TEST
3525 015244 104414 TBITR ;RESTORE T-BIT IF IT WAS ON
3526 015246 032777 004000 163662 BIT #BIT11,@SWR ;SEE IF THIS IS AN 11/24 WITH UB MEMORY ONLY
3527 015254 001405 BEQ 1$ ;BRANCH TO SETUP IF NOT
3528 015256 062737 000005 001100 ADD #5,$STSNM ;FUDGE $STSNM 5 TESTS AHEAD AND
3529 015264 000137 015744 JMP TST21 ;JUMP OVER NEXT 5 TESTS
3530
3531                                     :*****
3532                                     :
3533                                     : THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
3534                                     : USABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K. IN THE
3535                                     : DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
3536                                     : THIS WILL BE MAP REGISTERS 1 AND 2. MAP REGISTER 1 WILL POINT
3537                                     : TO PHYSICAL 4K - 8K SO "ACT-11" WILL WORK PROPERLY AND MAP
3538                                     : REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K. THIS MEANS THAT
3539                                     : KIPARO SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
3540                                     : ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
3541                                     : 020000 TO 037776 ON THE UNIBUS.
3542                                     :
3543                                     :*****
3544
3545 015270 013701 001302 1$: MOV LREGU,R1 ;PUT POINTER TO LOWEST MAP REGISTER IN R1
3546 015274 005721 TST (R1)+ ;POINT TO LOWER 16 BITS OF LOWEST + 1
3547 015276 012721 020000 MOV #20000,(R1)+ ;LOAD LOWER 16 BITS OF (LOWEST + 1), POINTING TO 4-8K
3548 015302 005021 CLR (R1)+ ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
3549 015304 005021 CLR (R1)+ ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
3550 015306 005021 CLR (R1)+ ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
3551 015310 013701 001256 MOV LOWEST,R1 ;LOAD R1
3552 015314 062701 000200 ADD #200,R1 ;POINT R1 TO (LOWEST + 1)
3553 015320 010137 172342 MOV R1,KIPAR1 ;LOAD PAR1
3554 015324 062701 000200 ADD #200,R1 ;POINT R1 TO (LOWEST + 2)
3555 015330 010137 172340 MOV R1,KIPARO ;LOAD PARO

```

3567

.SBTTL TEST # 14 - MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.

*TEST 14 MAIN MEM. T.O. THROUGH MAP, CODE RUN OVER U.B.

*:

* THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
* TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY. IT USES THE LOWEST
* USABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
* ZERO.

* THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THROUGH
* THE UNIBUS MAP.

*:

TST14:

	015334			
	015334	000004		
	015336	004737	005166	
	015342	015362	015354	000014
3568	015350	005037	004312	
3569	015354	004737	020014	
3570	015360	104015		

	SCOPE		
	JSR	PC,PRETST	;GO SET UP PRETEST DATA
	.WORD	TST15,20\$,14	;DATA USED BY PRETST
	CLR	LOEFLG	;CLEAR LOOP ON ERROR FLAG
20\$:	JSR	PC,MMTOTM	;GO DO TEST ON PAGE 60 OF THIS LISTING
	ERROR	+15	;RETURN HERE IF ERROR - UNIBUS DID NOT TIME OUT

3584

```
.SBTTL TEST # 15 - RELOC USING LOWEST USABLE MAP REG USING MAP REG
*****
*TEST 15 RELOC USING LOWEST USABLE MAP REG USING MAP REG
*
* THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
* MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
* IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
* THIS TEST WILL USE THE LOWEST USABLE MAP REGISTER.
* IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
* THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
*
* THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THROUGH
* THE UNIBUS MAP.
*
*****
```

```
015362
015362 000004
015364 004737 005166
015370 015420 015404 000015
3585 015376 012737 060000 060000
3586 015404 005037 004312
3587 015410 004737 004106
3588 015414 104211
3589 015416 000774
```

```
TST15:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST16,20$,15 ;DATA USED BY PRETST
MOV #060000,060000 ;MAKE SURE ADDRESS 060000 CONTAINS ITS OWN ADDRESS AS DATA
20$: CLR LOEFLG ;CLEAR LOEFLG - USED IN ERROR RETURN
1$: JSR PC,MAPADD ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE FOR ERROR
BR 1$ ;RETURN TO THE SUBROUTINE
```

3598

```
.SBTTL TEST # 16 - CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
:*****
:*TEST 16 CARRY PROP OF MAP'S RELOC ADDER USING MAP REG
:*
:* EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
:* WITH 00030000 UP TO 17000000. THAT IS THE FIRST OF EVERY 2K
:* WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
:* WORKING PROPERLY .
:*****
```

```
015420
015420 000004
015422 004737 005166
015426 015450 015434 000016
3599 015434 005037 004312
3600 015440 004737 004416
3601 015444 104211
3602 015446 000774
```

```
TST16:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST17,20$,16 ;DATA USED BY PRETST
CLR LOEFLG ;CLEAR LOEFLG - USED AS ERROR RETURN FLAG
JSR PC,TCPMRA ;GO DO THE TEST
ERROR +211 ;RETURN IS HERE IF AN ERROR
BR 1$ ;RETURN TO THE TEST
```

3633

```
.SBTTL TEST # 17 - VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
*****
:TEST 17 VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
*****
:
:*****NOTE*****
: THE MAP WILL BE SHUT OFF FOR THE REMAINDER OF THE DIAGNOSTIC
: BEFORE ANY TEST CODE IS EXECUTED. IT IS NOT NEEDED.
:*****
:*****
: THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
: SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
: OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
: THEN BIT 08 OF $$WREG IS SET TO 1 THROUGH APT SCRIPTING.
:
: THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE
: TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE INTERRUPT
: IS BEING CALLED FOR(CACHE PE INTR L).
:
: THIS TEST ASSUMES THAT ALL MODULES EXCEPT THE UBI MODULE ARE KNOWN
: GOOD MODULES.
:
: THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
: HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
: VERIFY TESTING OF THE UBI MODULE.
:
: TEST DESCRIPTION:
: VERIFY INTERRUPT LOGIC BY ASSURING THAT A TRAP OCCURS TO LOCATION
: 114 WHEN A LOCATION PREVIOUSLY WRITTEN
: WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
: CONDITIONS: PEA=0
: DCPI=0
:*****
```

```
TST17:
015450 000004
015450 004737 005166
015456 015610 015554 000017
3634 015464 005037 177572
3635 015470 005037 172516
3636 015474 005037 172340
3637 015500 012737 000200 172342
3638 015506 105737 007140
3639 015512 001403
3640 015514 105237 001100
3641 015520 000511
3642 015522 132737 000200 020163 1$:
3643 015530 001405
3644 015532 032737 000100 020164
3645 015540 001423
3646 015542 000404
3647 015544 032777 000100 163364 2$:
3648 015552 001416
3649 015554 012737 000005 005460 20$:
3650 015562 112737 000001 005470
3651 015570 004737 005324
3652 015574 012737 000000 177746
3653 015602 005703
3654 015604 001401

SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST20,20$,17 ;DATA USED BY PRETST
CLR MMR0 ;TURN OFF MEMORY MANAGEMENT
CLR MMR3 ;TURN OFF MAP RELOCATION
CLR KIPAR0 ;PUT PAR0 BACK WHERE IT SHOULD AND
MOV #200,KIPAR1 ;PUT PAR1 BACK WHERE IT SHOULD
TSTB CPUTYP ;IS THIS AN 11/24?
BEQ 1$ ;BRANCH OVER JUMP IF 11/44
INCB $TSTNM ;INCREMENT TEST NUMBER TO SIMULATE SKIPPING TEST
BR TST21 ;BRANCH OVER THIS AND NEXT TESTS
BITB #200,$ENVM ;IS APT SIZING?
BEQ 2$ ;NO: TRY HARDWARE SWITCH REGISTER
BIT #100,$SWREG ;YES APT IS SIZING; DOES APT SAY TO DO THIS TEST
BEQ TST20 ;:NO - SKIP TEST
BR 20$ ;YES, DO TEST
BIT #100,@SWR ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
BEQ TST20 ;:NO - SKIP TEST
MOV #5,CASH1 ;MOVE BYTE TO LOAD TO CACHE TO SUB ;DPM002
MOVB #1,CASH2 ;MOVE TEST 1 TO LOCATION IN SUBROUTINE ;DPM002
JSR PC,CASHSR ;GO DO THE CASH TEST
MOV #0,CACHE ;TURN CACHE ON
TST R3 ;DID TRAP OCCUR?
BEQ TST20 ;:BRANCH TO NEXT TEST IF YES
```

3655 015606 104020

ERROR +20

:INTERRUPT/ABORT LOGIC TESTS TRAP TO LOC 114 DID NOT OCCUR

3689

```
.SBTTL TEST # 20 - VERIFY TRAP DUE TO CACHE PARITY ABORT
*****
*TEST 20 VERIFY TRAP DUE TO CACHE PARITY ABORT
* THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
* SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
* OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
* THEN BIT 08 OF $SWREG IS SET TO 1 THROUGH APT SCRIPTING.
*
* THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE(BUS PBL)
* TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE ABORT
* IS BEING CALLED FOR.
*
* THIS TEST ASSUMES THAT ALL MODULES EXCEPT UBI ARE KNOWN GOOD MODULES
*
* THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
* HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
* VERIFY TESTING OF THE UBI MODULE.
*
* TEST DESCRIPTION:
*
* VERIFY ABORT LOGIC BY THE FOLLOWING RESULTS WHEN A LOCATION
* PREVIOUSLY WRITTEN WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
* 1. INSTRUCTION CYCLE WILL BE ABORTED
* 2. THE ABORT CAUSES TRAP TO 114
*
* PROCEDURE: INHIBIT CLOCKING OF PARITY ERROR SIGNAL TO
* INTERRUPT LOGIC. ALLOW CMPE<15> TO BE SET
* BY ABORT SIGNAL WHICH IS ASSERTED BY PARITY
* ERROR SIGNAL TO ABORT LOGIC.
*
* CONDITIONS: PEA=1
* DCPI=1
*****
```

```
015610
015610 000004
015612 004737 005166
015616 015744 015666 000020
3690 015624 032777 000100 163304
3691 015632 001437
3692 015634 132737 000200 020163
3693 015642 001405
3694 015644 032737 000100 020164
3695 015652 001427
3696 015654 000404
3697 015656 032777 000100 163252 12$:
3698 015664 001422
3699 015666 012737 000004 005460 20$:
3700 015674 112737 000002 005470
3701 015702 004737 005324
3702 015706 022704 177777
3703 015712 001401
3704 015714 104021
3705
3706 015716 005703 1$:
3707 015720 001401
```

```
TST20: SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST21,20$,20 ;DATA USED BY PRETST
BIT #BIT6,@SWR ;DOES SWITCH REGISTER SAY TO DO TEST?
BEQ 21$ ;NO - SKIP TEST
BITB #200,$ENVM ;IS APT SIZING?
BEQ 12$ ;NO;TRY HARDWARE SWITCH REGISTER
BIT #100,$SWREG ;YES APT IS SIZING;DOES APT SAY TO DO THIS TEST
BEQ 21$ ;NO - SKIP TEST
BR 20$ ;YES,DO TEST
BIT #100,@SWR ;DOES HARDWARE SWITCH REGISTER SAY TO DO TEST?
BEQ 21$ ;NO - SKIP TEST
MOV #4,CASH1 ;MOVE CACHE LOAD VALUE TO LOC IN SUB ;DPM002
MOVB #2,CASH2 ;MOVE 2ND TEST FLAG TO LOCATION IN SUB ;DPM002
JSR PC,CASHSR ;GO DO CACHE TEST
CMP #-1,R4 ;WAS INSTRUCTION ABORTED LEAVING R4 INTACT?
BEQ 1$ ;YES
ERROR +21 ;INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH
;DATA INDICATING THAT INSTRUCTION WAS NOT ABORTED
1$: TST R3 ;DID TRAP OCCUR
BEQ 2$ ;YES, PASS
```

3708	015722	104022				ERROR	+22		: INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT
3709	015724	012737	000000	177746	2\$:	MOV	#0,CACHE		: TURN CACHE ON
3710	015732	062737	000007	001100	21\$:	ADD	#7,\$STSNM		: ADD 7 TO \$STSNM TO COMPENSATE FOR 7 TESTS SKIPPED
3711	015740	000137	017164			JMP	TST30		: JUMP OVER NEXT 7 TESTS - THEY ARE FOR AN 11/24 ONLY

3724

.SBTTL TEST # 21 - LMA REGISTER PHYSICAL ADDRESS CHECK

*TEST 21 LMA REGISTER PHYSICAL ADDRESS CHECK

*
*

THE NEXT 7 TESTS ARE EXECUTED ON THE 11/24 ONLY.

*
*

THIS TEST IS TO CHECK OUT THE LMA (LAST MAPPED ADDRESS) REGISTER FOR PROPER CONTENTS. FIRST, THE PAR AND MAP REGISTERS ARE SET, THEN A PHYSICAL ADDRESS IS LOADED INTO AN EXPECTED DATA LOCATION. THEN THE MAP IS INSURED TO BE ON AND A MEMORY ACCESS IS DONE, USING THE MAP REGISTER SO THE LMA IS LOADED. THE LMA IS THEN CHECKED FOR CONTAINING THE PROPER CONTENTS, CALLING AN ERROR IF EXPECTED DATA DID NOT APPEAR.

*
*

TST21:

015744	000004			SCOPE		
015744	004737	005166		JSR	PC,PRETST	:GO SET UP PRETEST DATA
015752	016220	016104	000021	.WORD	TST22,20\$,21	:DATA USED BY PRETST
3725	015760	000416		BR	10\$:BRANCH OVER TEST PATTERN TABLE :DPM002
3726		000002		.RADIX	2	:SET RADIX TO BINARY REPRESENTATION :DPM002
3727	015762	040000		100\$: .WORD	0100000000000000	:SELECT MAP REG 2, ALL BITS CLEAR :DPM002
3728	015764	057776		.WORD	0101111111111110	:SELECT MAP REG 2, ALL BITS SET EXC 0 :DPM002
3729	015766	052524		.WORD	0101010101010100	:SELECT MAP REG 2, EVEN BITS SET :DPM002
3730	015770	045252		.WORD	0100101010101010	:SELECT MAP REG 2, ODD BITS SET :DPM002
3731	015772	043146		.WORD	0100011001100110	:SELECT MAP REG 2, PAIRS OF BITS SET :DPM002
3732	015774	057036		.WORD	0101111000011110	:SELECT MAP REG 2, 2 SETS OF 4 BITS SET :DPM002
3733	015776	040776		.WORD	0100000111111110	:SELECT MAP REG 2, 8 BITS SET :DPM002
3734				:EXPECTED RESULTS IN LMALOW:		
3735	016000	000000		.WORD	0000000000000000	:EXPECTED RESULT WITH ALL BITS CLEAR :DPM002
3736	016002	017776		.WORD	0001111111111110	:EXPECTED RESULT WITH ALL BITS 1 EXC 0 :DPM002
3737	016004	012524		.WORD	0001010101010100	:EXPECTED RESULT WITH EVEN BITS SET :DPM002
3738	016006	005252		.WORD	0000101010101010	:EXPECTED RESULT WITH ODD BITS SET :DPM002
3739	016010	003146		.WORD	0000011001100110	:EXPECTED RESULT WITH PAIRS OF BITS SET :DPM002
3740	016012	017036		.WORD	0001111000011110	:EXPECTED RESULT WITH GRPS OF 4 BITS =1 :DPM002
3741	016014	000776		.WORD	0000000111111110	:EXPECTED RESULT WITH 8 BITS SET :DPM002
3742		000010		.RADIX	8	:SET RADIX BACK TO OCTAL :DPM002
3743	016016	012701	015762	10\$: MOV	#100\$,R1	:MOVE TABLE ADDRESS TO R1 :DPM002
3744	016022	012702	000007	MOV	#7,R2	:DO 7 PATTERNS :DPM002
3745	016026	042737	000001	BIC	#BIT0,MMR0	:TURN OFF MEMORY MANAGEMENT
3746	016034	042737	000060	BIC	#60,MMR3	:TURN OFF 22-BIT AND MAP RELOCATION
3747	016042	013746	172344	MOV	KIPAR2,-(SP)	:SAVE KIPAR2
3748	016046	013737	001256	MOV	LOWEST,KIPAR2	:PUT PAR VALUE IN PAR2 TO ACCESS LOWEST MAP REGISTER
3749	016054	052737	000060	BIS	#60,MMR3	:TURN ON 22-BIT AND MAP RELOCATION
3750	016062	005037	172340	CLR	KIPAR0	:CLEAR PAR0 FOR PAGE ACCESSING THIS TEST
3751	016066	052737	000001	BIS	#BIT0,MMR0	:TURN ON MEMORY MANAGEMENT
3752	016074	005077	163200	CLR	@LREGL	:CLEAR LOWEST USEABLE MAP REGISTER LOW WORD
3753	016100	005077	163176	CLR	@LREGU	:CLEAR LOWEST USEABLE MAP REGISTER HIGH WORD
3754	016104	017137	000000	20\$: MOV	@0(R1),\$TMP0	:DO THE MAP REGISTER READ THRU THE MAP :DPM002
3755	016112	026137	000016	CMP	16(R1),LMALOW	:SEE IF LMA LWR 16 WERE LOADED PROPERLY :DPM002
3756	016120	001007		BNE	1\$:BRANCH TO CALL ERROR IF NOT :DPM002
3757	016122	013737	177736	MOV	LMAHI,\$TMP0	:MOVE HI 6 BITS TO \$TMP0 FOR PREPARATION
3758	016130	042737	177700	BIC	#177700,\$TMP0	:CLEAR ALL BUT LOWER 6 BITS
3759	016136	001424		BEQ	2\$:BRANCH AROUND ERROR IF OK
3760	016140	013737	177734	1\$: MOV	LMALOW,EADRS2	:MOVE LOWER 16 BITS OF RECEIVED DATA TO EADRS2 FOR ERROR
3761	016146	013737	177736	MOV	LMAHI,EADRS2+2	:MOVE UPPER 6 BITS OF RECEIVED DATA TO EADRS2+2 FOR ERROR
3762	016154	016137	000016	MOV	16(R1),EADRES	:MOVE EXPECTED DATA TO EADRES :DPM002
3763	016162	005037	005766	CLR	EADRES+2	:CLEAR THE UPPER LOCATION

3764	016166	011637	172344		MOV	(SP),KIPAR2	:RESTORE KIPAR2	
3765	016172	104023			ERROR	+23	:LMA NOT LOADED PROPERLY	
3766	016174	013737	001256	172344	MOV	LOWEST,KIPAR2	:RESET PAR2 TO PRE-ERROR SETUP	
3767	016202	012703	004000		MOV	#4000,R3	:MOVE 4000 TO WAIT LOOF COUNTER	
3768	016206	077301		15\$:	SOB	R3,15\$:WAIT A LITTLE	
3769	016210	005721		2\$:	TST	(R1)+	:INCREMENT TO NEXT TABLE VALUE	:DPM002
3770	016212	077244			SOB	R2,20\$:SUBTRACT 1 AND BRANCH IF NOT DONE YET	:DPM002
3771	016214	012637	172344	3\$:	MOV	(SP)+,KIPAR2	:RESTORE KIPAR2 FOR GOOD NOW	

3778

```
.SBTTL TEST # 22 - LMA FORCE JUMPER BIT TEST
:*****
:*TEST 22      LMA FORCE JUMPER BIT TEST
:*
:*      THIS TEST DETERMINES THAT THE FORCE JUMPER BIT OF THE LMA IS ZERO AFTER
:*      A SYSTEM RESET.
:*
:*****
```

016220	000004		
016220	004737	005166	
016222	016266	016234	000022
3779	000100		
3780	016234	042737	000060 172516
3781	016242	000005	
3782	016244	032737	000100 177736
3783	016252	001405	
3784	016254	013701	177736
3785	016260	042701	000100
3786	016264	104024	

```
TST22:
SCOPE
JSR      PC,PRETST      ;GO SET UP PRETEST DATA
        .WORD      TST23,20$,22 ;DATA USED BY PRETST
        =100        ;FORCE JUMPER BIT IS BIT 6
FJBIT   BIC      #60,MMR3 ;TURN OFF 22-BIT AND MAP RELOCATION
20$:    RESET     ;RESET THE WORLD, CLEARING THE FJBIT
        BIT      #FJBIT,LMAHI ;CHECK THE BIT FOR BEING ZERO
        BEQ      TST23    ;:BRANCH TO NEXT TEST IF OK
        MOV      LMAHI,R1 ;MOVE LMAHI TO R1 FOR ERROR CALL
        BIC      #FJBIT,R1 ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
        ERROR    +24     ;LMA FORCE JUMPER BIT NOT ZERO
```

3796

.SBTTL TEST # 23 - SETTING LMA FORCE JUMPER BIT TEST

*TEST 23 SETTING LMA FORCE JUMPER BIT TEST

*

* THIS TEST SETS THE FORCE JUMPER BIT AND TESTS ITS FUNCTIONALITY IF
 * THE JUMPERS ARE NOT IN THEIR DEFAULT STATE. IF NOT ('LOWEST' OR
 * 'HIGEST' DO NOT CONTAIN THE DEFAULT VALUES OF 170000 OR 177400
 * RESPECTIVELY), THIS TEST INSURES THAT THE PREVIOUSLY DISABLED MAP
 * REGISTERS ARE ENABLED WITH THE FJ BIT SET.

*

TST23:

016266	000004					SCOPE		
016266	004737	005166				JSR	PC,PRETST	:GO SET UP PRETEST DATA
016270	016474	016302	000023			.WORD	TST24,20\$,23	:DATA USED BY PRETST
3797	016302	052737	000100	177736	20\$:	BIS	#FJBIT,LMAHI	:SET THE BIT
3798	016310	032737	000100	177736		BIT	#FJBIT,LMAHI	:SEE IF IT WAS SET
3799	016316	001005				BNE	1\$:BRANCH IF SET
3800	016320	013701	177736			MOV	LMAHI,R1	:MOVE LMAHI TO R1 FOR ERROR CALL
3801	016324	052701	000100			BIS	#FJBIT,R1	:SET THE BIT THAT SHOULD HAVE BEEN SET
3802	016330	104025				ERROR	+25	:LMA FORCE JUMPER BIT NOT SET
3803	016332	022737	170000	001256	1\$:	CMP	#170000,LOWEST	:SEE IF MAP REGISTER 0 IS LOWEST
3804	016340	001004				BNE	2\$:BRANCH AROUND UPPER LIMIT CHECK IF NOT
3805	016342	022737	177400	001260		CMP	#177400,HIGEST	:SEE IF MAP REGISTER 31 IS HIGEST
3806	016350	001451				BEQ	TST24	:BRANCH TO NEXT TEST IF SO
3807	016352	052737	000060	172516	2\$:	BIS	#60,MMR3	:TURN ON 22-BIT AND MAP RELOCATION
3808	016360	012737	016440	001106		MOV	#6\$,SLPERR	:RESET LOOP ON ERROR TO 6\$
3809	016366	013746	172350			MOV	KIPAR4,-(SP)	:SAVE PAR4
3810	016372	013737	001262	172350		MOV	UBMLOW,KIPAR4	:MOVE LOWEST PAGE OF MEMORY WINDOW TO PAR4
3811	016400	012700	117776			MOV	#117776,R0	:THIS WILL BE USED TO SELECT PAR4, ADDRESS 17776
3812	016404	012702	125252			MOV	#125252,R2	:LOAD CONSTANT TO R2
3813	016410	000407				BR	5\$:BRANCH OVER LOOP SETUP
3814	016412	062737	000200	172350	4\$:	ADD	#200,KIPAR4	:MAP TO NEXT REGISTER
3815	016420	023737	001264	172350		CMP	UBMHI,KIPAR4	:SEE IF HIGEST HAS BEEN REACHED
3816	016426	001415				BEQ	9\$:BRANCH TO RESTORE PAR4 AND LEAVE TEST IF SO
3817	016430	004737	003474		5\$:	JSR	PC,TSTLOC	:GO TEST LOCATION FOR WRITEABILITY
3818	016434	000766				BR	4\$:BRANCH BACK FOR ANOTHER TEST IF LOCATION LOADED
3819	016436	000403				BR	7\$:BRANCH AROUND LOOP ON ERROR SETUP
3820	016440	004737	003474		6\$:	JSR	PC,TSTLOC	:GO TEST LOCATION FOR WRITEABILITY
3821	016444	000401				BR	8\$:GO TEST FOR LOOP ON ERROR IF OK NOW
3822	016446	104027			7\$:	ERROR	+27	:FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEF
3823	016450	032777	001000	162460	8\$:	BIT	#BIT9,aSWR	:SEE IF LOOP ON ERROR IS STILL SET
3824	016456	001370				BNE	6\$:BRANCH BACK TO LOOP SECTION IF SO
3825	016460	000754				BR	4\$:BRANCH BACK FOR ANOTHER TEST
3826	016462	012637	172350		9\$:	MOV	(SP)+,KIPAR4	:RESTORE KIPAR4
3827	016466	042737	000060	172516		BIC	#60,MMR3	

3833

```
.SBTTL TEST # 24 - CLEARING THE FORCE JUMPER BIT
:*****
:*TEST 24            CLEARING THE FORCE JUMPER BIT
:*
:*            THIS TEST CLEARS THE FJ BIT AND INSURES THAT IT IS SUCCESSFULLY CLEARED.
:*
:*****
```

```
016474
016474 000004
016476 004737 005166
016502 016540 016510 000024
3834 016510 042737 000100 177736
3835 016516 032737 000100 177736
3836 016524 001405
3837 016526 013701 177736
3838 016532 042701 000100
3839 016536 104024
```

```
TST24:
SCOPE
JSR    PC,PRETST            ;GO SET UP PRETEST DATA
      .TWORD    TST25,20$,24 ;DATA USED BY PRETST
BIC    #FJBIT,LMAHI        ;CLEAR THE BIT
BIT    #FJBIT,LMAHI        ;CHECK TO SEE THAT IT WAS CLEARED
BEQ    TST25                ;BRANCH IF CLEARED
MOV    LMAHI,R1            ;MOVE LMAHI TO R1 FOR ERROR CALL
BIC    #FJBIT,R1            ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
ERROR +24                 ;LMA FORCE JUMPER BIT NOT ZERO
```

3846

```
.SBTTL TEST # 25 - LMA CONTROL BITS TEST - DATI
:*****
:*TEST 25      LMA CONTROL BITS TEST - DATI
:*
:*      THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
:*      DATI.
:*
:*****
```

TST25:

016540	000004				SCOPE	
016540	004737	005166			JSR	PC,PRETST ;GO SET UP PRETEST DATA
016542	016702	016644	000025		.WORD	TST26,20\$,25 ;DATA USED BY PRETST
3847 016546	013746	172344			MOV	KIPAR2,-(SP) ;SAVE KIPAR2
3848 016554	012737	170000	172344		MOV	#170000,KIPAR2 ;LOAD KIPAR2
3849 016560	012737	000001	177572		MOV	#1,MMR0 ;TURN ON MEMORY MANAGEMENT
3850 016566	012737	000020	172516		MOV	#20,MMR3 ;TURN ON 22-BIT ADDRESSING
3851 016574	012700	056664			MOV	#1\$+BIT14,R0 ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0
3852 016602	011001				MOV	(R0),R1 ;DO A DATI
3853 016606	013737	177736	001174		MOV	LMAHI,\$TMP0 ;MOVE LMAHI TO \$TMP0 FOR PREPARATION OF EXPECTED
3854 016610	013737	001174	001200		MOV	\$TMP0,\$TMP2 ;MOVE IT TO \$TMP2 ALSO
3855 016616	042737	037777	001200		BIC	#37777,\$TMP2 ;CLEAR ALL BUT THE CONTROL BITS
3856 016624	023737	001174	001200		CMP	\$TMP0,\$TMP2 ;SEE IF EXPECTED DATA CAME UP
3857 016632	001416				BEQ	3\$;BRANCH TO FINISH TEST IF ALL CLEAR
3858 016640	000410				BR	1\$;BRANCH OVER LOOP ON ERROR SETUP
3859 016642	011001			20\$:	MOV	(R0),R1 ;DO A DATI
3860 016644	013737	177736	001174		MOV	LMAHI,\$TMP0 ;MOVE LMAHI TO \$TMP0 FOR COMPARE
3861 016646	023737	001174	001200		CMP	\$TMP0,\$TMP2 ;SEE IF EXPECTED DATA CAME UP
3862 016654	001401				BEQ	2\$;BRANCH AROUND ERROR IF IT DID
3863 016662	104026			1\$:	ERROR	+26 ;LMA CONTROL BITS INCORRECT
3864 016664	032777	001000	162242	2\$:	BIT	#BIT9,\$SWR ;SEE IF LOOP ON ERROR IS SET
3865 016666	001363				BNE	20\$;BRANCH BACK FOR ANOTHER TRY IF SET
3866 016674	012637	172344		3\$:	MOV	(SP)+,KIPAR2 ;RESTORE KIPAR2

3873 .SBTTL TEST # 26 - LMA CONTROL BITS TEST - DATO
 :*****
 :*TEST 26 LMA CONTROL BITS TEST - DATO
 :*
 :* THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A
 :* DATO.
 :*
 :*****

```

016702
016702 000004
016704 004737 005166
016710 017036 017000 000026
3874      100000
3875 016716 013746 172344
3876 016722 012737 170000 172344
3877 016730 012700 057020
3878 016734 010110
3879 016736 013737 177736 001174
3880 016744 013737 001174 001200
3881 016752 052737 100000 001200
3882 016760 042737 040000 001174
3883 016766 023737 001174 001200
3884 016774 001416
3885 016776 000410
3886 017000 010110
3887 017002 013737 177736 001174
3888 017010 023737 001174 001200
3889 017016 001401
3890 017020 104026
3891 017022 032777 001000 162106
3892 017030 001363
3893 017032 012637 172344
    
```

```

TST26:
SCOPE
JSR PC,PRETST ;GO SET UP PRETEST DATA
.WORD TST27,20$,26 ;DATA USED BY PRETST
=100000
DATO
MOV KIPAR2,-(SP) ;SAVE KIPAR2
MOV #170000,KIPAR2 ;LOAD KIPAR2
MOV #1$+BIT14,R0 ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0
MOV R1,(R0) ;DO A DATO
MOV LMAHI,$TMP0 ;MOVE LMAHI TO $TMP0 FOR CONTROL BIT ANALYSIS
MOV $TMP0,$TMP2 ;MOVE IT IT $TMP2 ALSO
BIS #DATO,$TMP2 ;SET AND CLEAR THE CONTROL BITS EXPECTED TO BE SET
BIC #BIT14,$TMP0 ;AND CLEAR SO $TMP2 WILL CONTAIN THE EXPECTED
CMP $TMP0,$TMP2 ;SEE IF BIT 15 IS SET AND 14 IS CLEAR
BEQ 3$ ;BRANCH IF OK
BR 1$ ;GO CALL ERROR
20$: MOV R1,(R0) ;DO A DATO
MOV LMAHI,$TMP0 ;MOVE LMA HIGH REGISTER CONTENTS TO $TMP0 FOR COMPARE
CMP $TMP0,$TMP2 ;SEE IF EXPECTED CAME UP
BEQ 2$ ;BRANCH AROUND ERROR IF IT DID
1$: ERROR +26 ;LMA CONTROL BITS INCORRECT
2$: BIT #BIT9,@SWR ;SEE IF LOOP ON ERROR IS SET
BNE 20$ ;BRANCH BACK FOR ANOTHER TRY IF SET
3$: MOV (SP)+,KIPAR2 ;RESTORE KIPAR2
    
```

3900

```
.SBTTL TEST # 27 - LMA CONTROL BITS TEST - DATOB  
:*****  
:TEST 27 LMA CONTROL BITS TEST - DATOB  
:  
: THIS TEST INSURES THE CONTROL BITS 14 AND 15 LOAD PROPERLY DOING A  
: DATOB.  
:  
:*****
```

```
017036  
017036 000004  
017040 004737 005166  
017044 017164 017126 000027  
3901 140000  
3902 017052 013746 172344  
3903 017056 012737 170000 172344  
3904 017064 012700 057146  
3905 017070 110110  
3906 017072 013737 177736 001174  
3907 017100 013737 001174 001200  
3908 017106 052737 140000 001200  
3909 017114 023737 001174 001200  
3910 017122 001416  
3911 017124 000410  
3912 017126 110110  
3913 017130 013737 177736 001174  
3914 017136 023737 001174 001200  
3915 017144 001401  
3916 017146 104026  
3917 017150 032777 001000 161760  
3918 017156 001363  
3919 017160 012637 172344  
3920  
3921  
3922  
3923  
3924  
3925  
3926
```

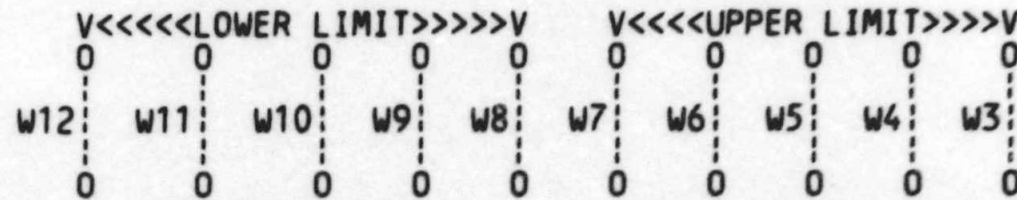
```
TST27:  
SCOPE  
JSR PC,PRETST ;GO SET UP PRETEST DATA  
.WORD TST30,20$,27 ;DATA USED BY PRETST  
=140000 ;DATOB CONTROL BITS STATUS=140000  
DATOB  
MOV KIPAR2,-(SP) ;SAVE KIPAR2  
MOV #170000,KIPAR2 ;LOAD KIPAR2  
MOV #1$+BIT14,R0 ;MOVE LOCATION ADDRESS +40000 (TO REF PAR2) TO R0  
MOVB R1,(R0) ;DO A DATOB  
MOV LMAHI,$TMP0 ;MOVE LMAHI TO $TMP0 FOR CONTROL BIT CHECK  
MOV $TMP0,$TMP2 ;MOVE TO $TMP2 ALSO  
BIS #DATOB,$TMP2 ;SET THE EXPECTED DATA BITS INTO $TMP0  
CMP $TMP0,$TMP2 ;SEE IF EXPECTED DATA CAME UP  
BEQ 3$ ;BRANCH IF OK  
BR 1$ ;BRANCH TO CALL ERROR  
20$: MOVB R1,(R0) ;DO A DATOB  
MOV LMAHI,$TMP0 ;MOVE LMA HIGH REGISTER CONTENTS TO $TMP0 FOR COMPARE  
CMP $TMP0,$TMP2 ;SEE IF EXPECTED DATA CAME UP  
BEQ 2$ ;BRANCH IF OK  
1$: ERROR +26 ;LMA CONTROL BITS INCORRECT  
2$: BIT #BIT9,@SWR ;SEE IF LOOP ON ERROR IS SET  
BNE 20$ ;BRANCH BACK FOR ANOTHER TRY IF SET  
3$: MOV (SP)+,KIPAR2 ;RESTORE KIPAR2  
:*****
```

```
>>NOTE<<: 'DATIP' CANNOT BE CHECKED IN THE 11/24 BECAUSE THE LMA IS  
WRITTEN TWICE WHEN A 'DATIP' IS EXECUTED, DESTROYING THE  
'DATIP' STATE THAT WAS WRITTEN FIRST.  
:*****
```


.SBTTL MEMORY ON UNIBUS TESTS HEADER

THE NEXT TWO TESTS WILL EXECUTE IF A '1' IN BIT 5 OF THE SWITCH REGISTER IS FOUND SET. IF IT HAS, IT THEN DETERMINES IF THERE IS ANY UNIBUS MEMORY - AN ERROR RESULTS IF THERE IS NONE. IF THERE IS MEMORY, IT THEN SIZES THE AMOUNT OF MEMORY ON THE UNIBUS, INFORMS THE USER HOW MUCH MEMORY IT FOUND ON THE FIRST PASS, THEN SETS AND CLEARS ALL BITS OF ALL LOCATIONS IN THE UNIBUS MEMORY FOUND USING THE 'MARCH' ALGORITHM.

M7098 FOR THE 11/44, M7134 FOR THE 11/24
JUMPER SETTINGS FOR THE MAP REGISTERS



*W3-W7 AND W8-W12 ARE THE BINARY-CODED PAGE NUMBER LIMIT (UPPER OR LOWER).
*A JUMPER IN CORRESPONDS TO A LOGIC '0'; A JUMPER OUT TO A LOGIC '1'. TO
*SET THE JUMPERS, DETERMINE WHICH UNIBUS PAGE THE MEMORY RESIDES IN (0 TO 31)
*BY CHECKING WHICH OF THE 5 ADDRESS BITS BA17-BA13 ARE ASSERTED. ALL ZEROS
*IS PAGE 0, 10000 IS PAGE 1, 01000 IS PAGE 2, 11000 IS PAGE 3, ETC. UP TO
*PAGE 31 (11111). FOR THE MEMORY TO BE DETECTED, IT MUST LIE AT OR ABOVE
*THE LOWER LIMIT JUMPER SETTING AND BELOW THE UPPER LIMIT JUMPER SETTING.
*THUS TO HAVE UNIBUS MEMORY IN PAGES 5-9, ONE WOULD SET THE LOWER LIMIT TO
*PAGE 5 (10100) OR W10 AND W12 OUT, AND W8, W9 AND W11 IN, AND THE UPPER LIMIT
*TO PAGE 10 (01010) OR W4 AND W6 OUT, AND W3, W5 AND W7 IN. UNIBUS MEMORY
*MUST BE CONTIGUOUS SINCE NO GAPS ARE PERMITTED.
:*****

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

3972

..SBTTL TEST # 30 - MEMORY ON UNIBUS TEST

 *TEST 30 MEMORY ON UNIBUS TEST

* THIS TEST FIRST CHECKS TO SEE IF THE UNIBUS MEMORY TESTS HAVE BEEN
 * SELECTED. IF NOT, THE EOP IS EXECUTED. IT THEN CHECKS FOR UNIBUS
 * MEMORY EXISTENCE BY CHECKING THAT THE CONTENTS OF LOWEST OR HIGHEST
 * LOCATIONS IS NOT IN ITS DEFAULT STATE. IF BOTH ARE, AN ERROR IS
 * CALLED AND THE EOP IS EXECUTED. IF EITHER ARE NOT, THE NEXT SECTION
 * IS EXECUTED THAT DETERMINES THE SIZE OF THE UB MEMORY AND TELLS
 * USER OF THE RESULTS ON THE FIRST PASS ONLY.

 TST30:

017164				
017164	000004			
017166	004737	005166		
017172	017412	017200	000030	
3973	017200	042737	000001	177572
3974	017206	032777	000040	161722
3975	017214	001425		
3976	017216	012737	021576	001362
3977	017224	022737	170000	001256
3978	017232	001020		
3979	017234	022737	177400	001260
3980	017242	001014		
3981	017244	012737	012744	001106
3982	017252	112737	000007	001100
3983	017260	112737	000007	020146
3984	017266	104017		
3985	017270	000137	021576	
3986	017274	005037	001320	
3987	017300	005037	001330	
3988	017304	005037	172516	
3989	017310	052737	000020	172516
3990	017316	005037	172340	
3991	017322	012737	000200	172342
3992	017330	013700	001274	
3993	017334	163700	001272	
3994	017340	005200		
3995	017342	010037	001316	
3996	017346	005046		
3997	017350	012746	000200	
3998	017354	013746	001262	
3999	017360	012746	000002	
4000	017364	005737	020150	
4001	017370	001010		
4002	017372	010046		
4003	017374	006316		
4004	017376	006316		
4005	017400	104401	006125	
4006	017404	104405		
4007	017406	104401	006174	
4008		021576		

SCOPE				
JSR	PC,PRETST			:GO SET UP PRETEST DATA
.WORD	TST31,20\$,30			:DATA USED BY PRETST
BIC	#1,MMR0			:TURN OFF MEMORY MANAGEMENT
BIT	#BITS,@SWR			:SEE IF THIS TEST HAS BEEN SELECTED
BEQ	2\$:BRANCH TO JUMP IF TEST NOT SELECTED
MOV	#\$EOP,NXTTST			:POINT TO ESCAPE VECTOR
CMP	#170000,LOWEST			:SEE IF LOWEST IS LOWEST
BNE	3\$:GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
CMP	#177400,HIGEST			:SEE IF HIGEST IS HIGEST
BNE	3\$:GO DO TEST IF IT ISN'T - UNIBUS MEMORY EXISTS
MOV	#SIZEJO,\$LPERR			:MOVE SIZE JUMPER ROUTINE TO LOOP ON ERROR
MOVB	#7,\$TSTNM			:EXECUTING AFTER TEST 7 ON LOOPBACK
MOVB	#7,\$TESTN			:EXECUTING AFTER TEST 7 ON LOOPBACK
ERROR	+17			:NO UNIBUS MEMORY EXISTS
JMP	\$EOP			:JUMP TO END OF PASS
CLR	ERRCNT			:CLEAR THE ERROR COUNT INDICATOR
CLR	PCPUER			:CLEAR THE ERROR REGISTER RECEIVER
CLR	MMR3			:CLEAR MEMORY MANAGEMENT REGISTER MMR3
BIS	#20,MMR3			:TURN ON 22-BIT MAPPING
CLR	KIPAR0			:MAP PAR0 TO 0-4K
MOV	#200,KIPAR1			:MAP PAR1 TO 4-8K
MOV	UBRHI,R0			:MOVE UBRHI TO R0
SUB	UBRLOW,R0			:SUBTRACT UBRLOW FROM IT, AND
INC	R0			:ADD LAST BLOCK OF 4K TO LOOP COUNTER
MOV	R0,NUMOFK			:SAVE R0 IN NUMOFK
CLR	-(SP)			:CLEAR THE MAJOR LOOP INDICATOR ON STACK
MOV	#200, -(SP)			:MOVE PAR CHANGE TO STACK
MOV	UBMLOW, -(SP)			:MOVE STARTING PAR VALUE TO STACK
MOV	#2, -(SP)			:MOVE INCREMENT VALUE TO STACK
TST	\$PASS			:SEE IF THIS IS FIRST PASS
BNE	TST31			:BRANCH TO NEXT TEST IF NOT
MOV	R0, -(SP)			:MOVE LOOP COUNTER TO THE STACK AND
ASL	(SP)			:ROTATE THIS TO THE LEFT 2 PLACES
ASL	(SP)			:TO INDICATE NUMBER OF K IN OCTAL
TYPE	,UBMAVA			:GO TYPE THE UNI-BUS MEMORY AVAILABLE MESSAGE
TYPDS				:GO TYPE THE NUMBER IN DECIMAL
TYPE	,UBMEND			:TYPE A " K" AND <CRLF>

TST32=\$EOP

4015

.SBTTL TEST # 31 - USING MARCH ALGORITHM, CHECK UB MEMORY

*TEST 31 USING MARCH ALGORITHM, CHECK UB MEMORY

* THIS TEST LOADS PATTERN 125252 INTO ALL LOCATIONS IN UB MEMORY, THEN
* USING THE MARCH ALGORITHM, CHECKS THE MEMORY

TST31:

017412	000004					SCOPE		
017412	004737	005166				JSR	PC,PRETST	:GO SET UP PRETEST DATA
017414	021576	017530	000031			.WORD	TST32,20\$,31	:DATA USED BY PRETST
4016 017420	012704	125252				MOV	#125252,R4	:MOVE FIRST TEST PATTERN TO R4
4017 017432	012705	052525				MOV	#52525,R5	:MOVE SECOND TEST PATTERN TO R5
4018 017436	013737	001262	172354	1\$:		MOV	UBMLOW,KIPAR6	:INITIALIZE PAR6
4019 017444	013700	001316				MOV	NUMOFK,R0	:REINITIALIZE LOOP COUNTER R0
4020 017450	052737	000001	177572			BIS	#1,MMR0	:TURN ON MEMORY MANAGEMENT
4021 017456	012702	010000		2\$:		MOV	#10000,R2	:ACCESS ALL WORDS IN THIS 4K BLOCK
4022 017462	012703	140000				MOV	#140000,R3	:FIRST ADDRESS OF THIS PAGE
4023 017466	010423			3\$:		MOV	R4,(R3)+	:MOVE THE PATTERN TO THE LOCATION
4024 017470	077202					SOB	R2,3\$:SUBTRACT 1 AND BRANCH IF 4K NOT DONE
4025 017472	062737	000200	172354			ADD	#200,KIPAR6	:MAP TO NEXT 4K BLOCK
4026 017500	077012					SOB	R0,2\$:SUBTRACT 1 AND BRANCH IF BLOCKS OF 4K NOT DONE
4027 017502	016637	000002	172354			MOV	2(SP),KIPAR6	:REINITIALIZE KIPAR6 TO POINT AT BEGINNING
4028 017510	010100					MOV	R1,R0	:REINITIALIZE LOOP COUNTER R0
4029 017512	012702	010000		4\$:		MOV	#10000,R2	:ACCESS ALL WORDS IN THIS 4K BLOCK
4030 017516	012703	140000				MOV	#140000,R3	:FIRST ADDRESS OF THIS PAGE
4031 017522	066603	000006				ADD	6(SP),R3	:ADD OFFSET FOR THIS MAJOR PASS
4032 017526	000401					BR	5\$:BRANCH OVER LOOP ON ERROR PREPARATION
4033 017530	010413			20\$:		MOV	R4,(R3)	:REWRITE 1ST PATTERN TO LOCATION FOR LOOP ON ERROR
4034 017532	020413			5\$:		CMP	R4,(R3)	:SEE IF IT WAS LOADED PROPERLY
4035 017534	001403					BEQ	6\$:BRANCH AROUND ERROR CALL IF OK
4036 017536	010437	001204				MOV	R4,\$TMP4	:MOVE EXPECTED DATA TO \$TMP4
4037 017542	000405					BR	7\$:GO COMPLETE DATA FETCHING AND CALL ERROR
4038 017544	005113			6\$:		COM	(R3)	:COMPLEMENT THAT LOCATION TO PRODUCE SECOND TEST PATTERN
4039 017546	020513					CMP	R5,(R3)	:SEE IF IT IS THE COMPLEMENT
4040 017550	001417					BEQ	11\$:BRANCH AROUND ERROR CALL IF IT IS
4041 017552	010537	001204				MOV	R5,\$TMP4	:MOVE EXPECTED DATA TO \$TMP4
4042 017556	011337	001206		7\$:		MOV	(R3),\$TMP5	:MOVE RECEIVED DATA TO \$TMP5
4043 017562	005737	001330				TST	PCPUER	:SEE IF THIS ACCESS TIMED OUT - IF IT DID, BRANCH
4044 017566	001010					BNE	11\$:AROUND ERROR CALLS - TIMEOUT ROUTINE LOGGED ERROR
4045 017570	010337	005764				MOV	R3,EADRES	:MOVE ADDRESS IN R3 TO EADRES FOR ERROR CALL
4046 017574	104206			8\$:		ERROR	+206	:DATA PATTERN NOT CORRECT
4047 017576	000404					BR	11\$:BRANCH AROUND INCREMENT AND CLEAR INSTRUCTIONS
4048 017600	005237	001110		9\$:		INC	\$ERTTL	:STILL COUNT THIS AS AN ERROR
4049 017604	005037	001330		10\$:		CLR	PCPUER	:CLEAR TIMEOUT RECEIVER
4050 017610	061603			11\$:		ADD	(SP),R3	:ADD INCREMENT/DECREMENT VALUE TO R3
4051 017612	077231					SOB	R2,5\$:SUBTRACT 1 AND BRANCH IF 4K NOT CHECKED
4052 017614	066637	000004	172354			ADD	4(SP),KIPAR6	:MAP TO NEXT 4K BLOCK
4053 017622	077045					SOB	R0,4\$:BRANCH BACK IF MORE BLOCKS TO CHECK
4054 017624	005766	000006				TST	6(SP)	:TEST TO SEE IF THIS IS SECOND PASS
4055 017630	001404					BEQ	12\$:BRANCH TO 2ND PASS SETUP IF NOT
4056 017632	062706	000010				ADD	#10,SP	:CLEAN UP STACK
4057 017636	000137	021576				JMP	\$EOP	:JUMP TO END OF PASS
4058 017642	012766	017776	000006	12\$:		MOV	#17776,6(SP)	:MOVE 2K WORDS -2 (ALSO 2ND PASS INDICATOR) TO STACK
4059 017650	012766	177600	000004			MOV	#-200,4(SP)	:MOVE REVERSE PAR STEP TO STACK
4060 017656	013766	001264	000002			MOV	UBMHI,2(SP)	:MOVE LAST PAR VALUE TO STACK

4061 017664 012716 177776
4062 017670 000662

MOV #-2.(SP)
BR 1\$

;MOVE DECREMENT VALUE TO STACK
;BRANCH BACK FOR SECOND PASS

```

4063      020000      . =20000      ;THE APT TABLES NEED TO START AT 20000 - THIS STATEMENT DOES THAT
4064
4065      177777      ADDW0= 177777
4066      177777      ADDW1= 177777
4067
.SBTTL   APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      020000      . $X= .      ;;SAVE CURRENT LOCATION
000024   000024      =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      000200      200      ;;FOR APT START UP
000044   000044      =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
      020000      $APTHDR ;;POINT TO APT HEADER BLOCK
      020000      =. $X      ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
$APTHD:
020000   000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
020002   020142      $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
020004   000005      $STMT: .WORD 5      ;;RUN TIM OF LONGEST TEST
020006   000010      $PASTM: .WORD 10     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
020010   000000      $UNITM: .WORD 0      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
020012   000052      .WORD SETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
    
```

```

4069 .SBTTL SUBROUTINE TO TEST TIMEOUT THROUGH UNIBUS MAP
4070 :*****
4071 020014 005737 004312 MMTOTM: TST LOEFLG ;SEE IF THIS ENTRY IS FROM ERROR LOOPING
4072 020020 001402 BEQ 1$ ;BRANCH IF NOT
4073 020022 062706 000002 ADD #2,SP ;CLEAN EXTRA RETURN OFF STACK
4074 020026 005037 001320 1$: CLR ERRCNT ;CLEAR ERRCNT FOR THIS TEST
4075 020032 013737 001256 172350 MOV LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
4076 020040 012777 000074 161234 MOV #74,@LREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
4077 020046 005077 161226 CLR @LREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
4078 020052 032777 004000 161056 BIT #BIT11,@SWR ;SEE IF AN 11/24 WITH UB MEMORY ONLY
4079 020060 001403 BEQ 20$ ;BRANCH IF NOT
4080 020062 012737 177600 172350 MOV #177600,KIPAR4 ;RESET KIPAR4 SO A TIMEOUT THROUGH MAP CAN BE EXPECTED
4081 020070 005037 001330 20$: CLR PCPUER ;CPU ERROR REGISTER LOCATION
4082 020074 012737 000020 001326 MOV #TIMOUT,CPUEXP ;EXPECTING TIMEOUT IN THIS TEST
4083 020102 013703 100000 MOV 100000,R3 ;TRY TO READ THROUGH PAGE 4 THIS REFERENCE WILL GO OUT
4084 :ON THE UNIBUS TO SELECT THE LOWEST USABLE MAP REGISTER (DEFAULT MAP REG. 0). PHYSICAL
4085 :ADDRESS 1770000 IS THEN GENERATED, WHICH SHOULD TIME OUT SINCE IT IS THE FIRST
4086 :NON-EXISTENT LOCATION.
4087 020106 005037 001326 CLR CPUEXP ;CLEAR LOCATION - NO MORE TIMEOUTS FOR A WHILE
4088 020112 022737 000020 001330 CMP #TIMOUT,PCPUER ;THE UNIBUS SHOULD HAVE TIMED OUT
4089 020120 001405 BEQ 3$ ;BRANCH IF CONDITION WAS CORRECT
4090 020122 011646 000001 004312 2$: MOV (SP),-(SP) ;PUSH ANOTHER RETURN ONTO THE STACK FOR POSSIBLE ERROR LOOP
4091 020124 012737 000001 004312 MOV #1,LOEFLG ;SET ERROR LOOP FLAG
4092 020132 000402 BR 4$ ;EXIT
4093 020134 062716 000002 3$: ADD #2,(SP) ;CORRECT RETURN PC OVER ERROR CALL
4094 020140 000207 4$: RTS PC ;EXIT

```

4095

```
.SBTTL APT MAILBOX-ETABLE
*****
.EVEN
020142 $MAIL: ::APT MAILBOX
020142 000000 $MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
020144 000000 $FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
020146 000000 $TESTN: .WORD ATESTN ::TEST NUMBER
020150 000000 $PASS: .WORD APASS ::PASS COUNT
020152 000000 $DEVCT: .WORD ADEVCT ::DEVICE COUNT
020154 000000 $UNIT: .WORD AUNIT ::I/O UNIT NUMBER
020156 000000 $MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
020160 000000 $MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
020162 $ETABLE: ::APT ENVIRONMENT TABLE
020162 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
020163 000 $ENVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
020164 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
020166 000000 $USWR: .WORD AUSWR ::USER SWITCHES
020170 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
*
* BIT 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
020172 000 $MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
020173 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
020174 000000 $MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
*
* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
020176 000 $MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
020177 000 $MTYP2: .BYTE AMTYP2 ::MEM. TYPE,BLK#2
020200 000000 $MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
020202 000 $MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
020203 000 $MTYP3: .BYTE AMTYP3 ::MEM. TYPE,BLK#3
020204 000000 $MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
020206 000 $MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
020207 000 $MTYP4: .BYTE AMTYP4 ::MEM. TYPE,BLK#4
020210 000000 $MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
020212 000000 $VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
020214 000000 $VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
020216 000000 $BASE: .WORD ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
020220 000000 $DEVN: .WORD ADEVN ::DEVICE MAP
020222 000000 $CDW1: .WORD ACDW1 ::CONTROLLER DESCRIPTION WORD#1
020224 000000 $CDW2: .WORD ACDW2 ::CONTROLLER DESCRIPTION WORD#2
020226 177777 $DDW0: .WORD ADDW0 ::DEVICE DESCRIPTOR WORD#0
020230 177777 $DDW1: .WORD ADDW1 ::DEVICE DESCRIPTOR WORD#1
020232 000000 $DDW2: .WORD ADDW2 ::DEVICE DESCRIPTOR WORD#2
020234 000000 $DDW3: .WORD ADDW3 ::DEVICE DESCRIPTOR WORD#3
020236 000000 $DDW4: .WORD ADDW4 ::DEVICE DESCRIPTOR WORD#4
020240 000000 $DDW5: .WORD ADDW5 ::DEVICE DESCRIPTOR WORD#5
020242 000000 $DDW6: .WORD ADDW6 ::DEVICE DESCRIPTOR WORD#6
020244 000000 $DDW7: .WORD ADDW7 ::DEVICE DESCRIPTOR WORD#7
020246 000000 $DDW8: .WORD ADDW8 ::DEVICE DESCRIPTOR WORD#8
020250 000000 $DDW9: .WORD ADDW9 ::DEVICE DESCRIPTOR WORD#9
```

020252 000000
020254 000000
020256 000000
020260 000000
020262 000000
020264 000000
020266

\$DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTOR WORD#10
\$DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTOR WORD#11
\$DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTOR WORD#12
\$DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTOR WORD#13
\$DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTOR WORD#14
\$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15
\$ETEND:

4097

.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<4:0>
:*CALL
:*          SCOPE          ;;SCOPE=IOT
    
```

```

020266          005037 001360          $SCOPE:
020266          005037 001320          CLR      RETRY          ;CLEAR RETRY FLAG AN THE START OF EACH TEST
020272          005037 001320          CLR      ERRCNT        ;CLEAR THE MULTIPLE ERROR COUNTER
020276          005037 001246          CLR      DATAOR       ;LOCATION FOR LOGICAL OR OF BAD DATA
020302          005037 001236          CLR      ADDROR        ;LOCATION FOR LOGICAL OR OF ADDRESS
020306          005037 001240          CLR      ADDROR+2      ;LOCATION FOR UPPER 6 BITS OF LOGICAL OR OF ADDRESS
020312          005037 001254          CLR      PATTOR        ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
020316          012737 177777 001242  MOV      #-1,DATAND    ;LOCATION FOR LOGICAL AND OF BAD DATA
020324          012737 177777 001232  MOV      #-1,ADRAND    ;LOCATION FOR LOGICAL AND OF ADDRESS
020332          012737 000077 001234  MOV      #77,ADRAND+2  ;LOCATION FOR UPPER 6 BITS OF LOGICAL AND OF ADDRESS
020340          012737 177777 001252  MOV      #-1,PATAND    ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
020346          012737 000077 005766  MOV      #77,EADRES+2  ;RESTORE UPPER 6 BIT LOCATION OF EADRES+2
020354          012737 000077 005772  MOV      #77,EADRS2+2 ;RESTORE UPPER 6 BIT LOCATION OF EADRS2+2
020362          032777 040000 160546  1$:     BIT      #BIT14,@SWR  ;;LOOP ON PRESENT TEST?
020370          001134          BNE      $OVER         ;;YES IF SW14=1
:#####START OF CODE FOR THE XOR TESTER#####
020372          000416          $XTSTR: BR      6$
020374          013746 000004          MOV      @ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
020400          012737 020420 000004  MOV      #5$,@ERRVEC  ;SET FOR TIMEOUT
020406          005737 177060          TST      @#177060     ;TIME OUT ON XOR?
020412          012637 000004          MOV      (SP)+,@ERRVEC ;RESTORE THE ERROR VECTOR
020416          000503          BR      $$VLAD        ;GO TO THE NEXT TEST
020420          022626          5$:     CMP      (SP)+,(SP)+  ;CLEAR THE STACK AFTER A TIME OUT
020422          012637 000004          MOV      (SP)+,@ERRVEC ;RESTORE THE ERROR VECTOR
020426          000443          BR      7$           ;LOOP ON THE PRESENT TEST
020430          6$:;#####END OF CODE FOR THE XOR TESTER#####
020430          032777 000400 160500  BIT      #BIT08,@SWR  ;;LOOP ON SPEC. TEST?
020436          001407          BEQ      2$          ;BR IF NO
020440          017746 160472          MOV      @SWR,-(SP)   ;SET DESIRED TEST NUM. FROM SWR
020444          042716 000340          BIC      #$$SRMK,(SP) ;STRIP AWAY UNDESIRED BITS
020450          122637 001100          CMPB    (SP)+,$TSTNM ;ON THE RIGHT TEST?
020454          001502          BEQ      $OVER       ;BR IF YES
020456          013737 177766 020700  2$:     MOV      177766,CPSAVE ;MOVE CPU ERR REG VALUE TO LOC FOR TST :DPM001
020464          032737 000001 020700  BIT      #BIT00,CPSAVE ;SEE IF THE POWER MONITOR BIT IS ON :DPM001
020472          001406          BEQ      2000$       ;BRANCH TO CONTINUE ROUTINE IF CLEAR :DPM001
020474          042737 000001 177766  BIC      #BIT00,177766 ;CLEAR THE BIT FOUND TO BE SET :DPM001
020502          104177          EMT      +177        ;CALL SPECIAL POWER FAIL BIT ERROR CALL :DPM001
020504          105037 001101          CLRB    $ERFLG       ;CLEAR THE ERROR FLAG :DPM001
020510          105737 001101          2000$: TSTB    $ERFLG      ;HAS AN ERROR OCCURRED?
020514          001421          BEQ      3$          ;BR IF NO
020516          123737 001113 001101  CMPB    $ERMAX,$ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
020524          101015          BHI     3$          ;BR IF NO
    
```

```

020526 032777 001000 160402      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
020534 001404      BEQ      4$              ;;BR IF NO
020536 013737 001106 001104 7$:   MOV      $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
020544 000446      BR              $OVER
020546 105037 001101      CLR      $ERFLG        ;;ZERO THE ERROR FLAG
020552 005037 001212      CLR      $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
020556 000415      BR              1$      ;;ESCAPE TO THE NEXT TEST
020560 032777 004000 160350 3$:   BIT      #BIT11,@SWR   ;;INHIBIT ITERATIONS?
020566 001011      BNE      1$           ;;BR IF YES
020570 005737 020150      TST      $PASS         ;;IF FIRST PASS OF PROGRAM
020574 001406      BEQ      1$           ;;      INHIBIT ITERATIONS
020576 005237 001102      INC      $ICNT         ;;INCREMENT ITERATION COUNT
020602 023737 001212 001102      CMP      $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
020610 002024      BGE      $OVER        ;;BR IF MORE ITERATION REQUIRED
020612 012737 000001 001102 1$:   MOV      #1,$ICNT      ;;REINITIALIZE THE ITERATION COUNTER
020620 013737 020676 001212      MOV      $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
020626 105237 001100      $SVLAD: INCB      $TSTNM ;;COUNT TEST NUMBERS
020632 113737 001100 020146      MOV      $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
020640 011637 001104      MOV      (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
020644 011637 001106      MOV      (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
020650 005037 001214      CLR      $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
020654 112737 000001 001113      MOV      #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
020662 013777 001100 160250 $OVER:  MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
020670 013716 001104      MOV      $LPADR,(SP)  ;;FUDGE RETURN ADDRESS
020674 000002      RTI
020676 000002      $MXCNT: 2.           ;;FIXES PS
020700 000000      CPSAVE: .WORD      0  ;;MAX. NUMBER OF ITERATIONS
                                ;;LOCATION TO SAVE CPU ERR REG CONTENTS ;DPM001

```

4099

.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
*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
    
```

```

020702 000000          IBSAVE: .WORD 0          ;LOC'N TO HOLD $ITEMB DURING DUAL ERR ;DPM001
020704 105037 020702  SERROR: CLR B IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION ;DPM001
020710 113737 001100 020146 MOVB $TSTNM,$TESTN ;SAVE TEST NUMBER FOR ERROR TYPE OUT
020716 005237 001320 INC ERRCNT          ;COUNT ALL MULTIPLE ERRORS
020722 010037 001160 MOV R0,$REG0        ;SAVE R0 FOR POSSIBLE TYPE OUT
020726 010137 001162 MOV R1,$REG1        ;SAVE R1 FOR POSSIBLE TYPE OUT
020732 010237 001164 MOV R2,$REG2        ;SAVE R2 FOR POSSIBLE TYPE OUT
020736 010337 001166 MOV R3,$REG3        ;SAVE R3 FOR POSSIBLE TYPE OUT
020742 010437 001170 MOV R4,$REG4        ;SAVE R4 FOR POSSIBLE TYPE OUT
020746 010537 001172 MOV R5,$REG5        ;SAVE R5 FOR POSSIBLE TYPE OUT
020752 105237 001101 7$: INCB $ERFLG          ;;SET THE ERROR FLAG
020756 001775 BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
020760 013777 001100 160152 MOV $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
020766 032777 002000 160142 BIT #BIT10,@SWR     ;;BELL ON ERROR?
020774 001402 BEQ 1$          ;;NO - SKIP
020776 104401 001216 TYPE ,SBELL          ;;RING BELL
021002 005237 001110 1$: INC $ERTTL          ;;COUNT THE NUMBER OF ERRORS
021006 011637 001114 MOV (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
021012 162737 000002 001114 SUB #2,$ERRPC
021020 117737 160070 001112 MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
021026 122737 000177 001112 CMPB #177,$ITEMB   ;SEE IF THIS IS THE POWER FAIL CALL ;DPM001
021034 001426 BEQ 2001$        ;BRANCH AROUND ROUTINE IF IT IS ;DPM001
021036 105737 020702 TSTB IBSAVE        ;SEE IF THIS IS THE 2ND ERROR CALL ;DPM001
021042 001021 BNE 2000$        ;BRANCH IF SO ;DPM001
021044 013737 177766 020700 MOV 177766,CPSAVE  ;MOVE CPU ERR REG TO CPSAVE FOR TEST ;DPM001
021052 032737 000001 020700 BIT #BIT00,CPSAVE  ;SEE IF POWER MONITOR BIT IS SET ;DPM001
021060 001414 BEQ 2001$        ;BRANCH IF OK ;DPM001
021062 042737 000001 177766 BIC #BIT00,177766 ;CLEAR THE BIT FOUND SET ;DPM001
021070 113737 001112 020702 MOVB $ITEMB,IBSAVE ;MAKE IBSAVE NON-ZERO FOR DUAL CALL ;DPM001
021076 112737 000177 001112 MOVB #177,$ITEMB   ;SET $ITEMB TO SPECIAL POWER FAIL PNTR ;DPM001
021104 000402 BR 2001$          ;BRANCH OVER IBSAVE CLEARING ;DPM001
021106 105037 020702 2000$: CLR B IBSAVE          ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT ;DPM001
021112 2001$:
021112 032777 020000 160016 BIT #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
021120 001004 BNE 20$          ;;SKIP TYPEOUTS
021122 004737 002026 JSR PC,ERTYPE      ;;GO TO USER ERROR ROUTINE
021126 104401 001223 TYPE ,$CRLF
021132 20$:
021132 122737 000001 020162 CMPB #APTENV,$ENV  ;;RUNNING IN APT MODE
021140 001007 BNE 2$          ;;NO,SKIP APT ERROR REPORT
021142 113737 001112 021154 MOVB $ITEMB,21$   ;;SET ITEM NUMBER AS ERROR NUMBER
021150 004737 021346 JSR PC,$ATY4      ;;REPORT FATAL ERROR TO APT
021154 000 21$: .BYTE 0
    
```

```

021155      000
021156      000777      22$: BR      22$      ;;APT ERROR LOOP
021160      105737      020702  2$: TSTB     IBSAVE    ;;SEE IF POWER FAIL ERROR CALL      ;DPM001
021164      001004      3$: BNE      3$      ;;BRANCH IF NOT - HALT NOT ALLOWED  ;DPM001
021166      005777      157744 TST      @SWR     ;;HALT ON ERROR
021172      100001      BPL      3$      ;;SKIP IF CONTINUE
021174      000000      HALT     ;;HALT ON ERROR!
021176      032777      001000  157732  3$: BIT      #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
021204      001405      4$: BEQ      4$      ;;BR IF NO
021206      105737      020702 TSTB     IBSAVE    ;;SEE IF THIS IS THE PWR MNTR BIT ERROR ;DPM001
021212      001257      7$: BNE      7$      ;;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
021214      013716      001106 MOV      $I.PERR,(SP) ;;FUDGE RETURN FOR LOOPING
021220      005737      001214  4$: TST      $ESCAPE  ;;CHECK FOR AN ESCAPE ADDRESS
021224      001405      5$: BEQ      5$      ;;BR IF NONE
021226      105737      020702 TSTB     IBSAVE    ;;SEE IF THIS IS THE PWR MNTR BIT ERROR ;DPM001
021232      001247      7$: BNE      7$      ;;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
021234      013716      001214 MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
021240
021240      022737      022024  000042  5$: CMP      #$ENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
021246      001001      6$: BNE      6$      ;;BRANCH IF NO
021250      000000      HALT     ;;YES
021252
021252      105737      020702  6$: TSTB     IBSAVE    ;;SEE IF THIS IS THE PWR FAIL ERROR CALL ;DPM001
021256      001235      7$: BNE      7$      ;;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;DPM001
021260      032777      001000  157650 BIT      #SW9,@SWR  ;;ARE WE LOOPING ON THIS ERROR?
021266      001417      1000$ BEQ      1000$     ;;BRANCH IF NOT
021270      012737      177777  177766 MOV      #-1,CPUERR ;;CLEAR CPU ERROR REGISTER
021276      042737      177776  177572 BIC      #177776,MMRO ;;CLEAR MEMORY MANAGEMENT STATUS REGISTER
021304      012737      177777  003032 MOV      #-1,TOFLAG ;;INITIALIZE TRAP FLAG
021312      012737      177777  003224 MOV      #-1,CPFLAG ;;INITIALIZE CP TRAP FLAG
021320      012737      177777  003374 MOV      #-1,MMFLAG ;;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
021326      000002      1000$: RTI      ;;RETURN TO TEST

```

4101

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
021330 112737 000001 021574 $ATY1:  MOVB  #1,$FFLG  ;;TO REPORT FATAL ERROR
021336 112737 000001 021572 $ATY3:  MOVB  #1,$MFLG  ;;TO TYPE A MESSAGE
021344 000403                BR      $ATYC
021346 112737 000001 021574 $ATY4:  MOVB  #1,$FFLG  ;;TO ONLY REPORT FATAL ERROR
021354 $ATYC:
021354 010046                MOV   R0,-(SP)  ;;PUSH R0 ON STACK
021356 010146                MOV   R1,-(SP)  ;;PUSH R1 ON STACK
021360 105737 021572                TSTB  $MFLG    ;;SHOULD TYPE A MESSAGE?
021364 001450                BEQ   5$      ;;IF NOT: BR
021366 122737 000001 020162        CMPB  #APTENV,$ENV  ;;OPERATING UNDER APT?
021374 001031                BNE   3$      ;;IF NOT: BR
021376 132737 000100 020163        BITB  #APTSPOOL,$ENVM  ;;SHOULD SPOOL MESSAGES?
021404 001425                BEQ   3$      ;;IF NOT: BR
021406 017600 000004                MOV   @4(SP),R0  ;;GET MESSAGE ADDR.
021412 062766 000002 000004        ADD   #2,4(SP)  ;;BUMP RETURN ADDR.
021420 005737 020142        1$:  TST   $MSGTYPE  ;;SEE IF DONE W/ LAST XMISSION?
021424 001375                BNE   1$      ;;IF NOT: WAIT
021426 010037 020156        MOV   R0,$MSGAD  ;;PUT ADDR IN MAILBOX
021432 105720        2$:  TSTB  (R0)+    ;;FIND END OF MESSAGE
021434 001376                BNE   2$
021436 163700 020156        SUB   $MSGAD,R0  ;;SUB START OF MESSAGE
021442 006200                ASR   R0        ;;GET MESSAGE LNGTH IN WORDS
021444 010037 020160        MOV   R0,$SMSGLGT  ;;PUT LENGTH IN MAILBOX
021450 012737 000004 020142        MOV   #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
021456 000413                BR    5$
021460 017637 000004 021504        3$:  MOV   @4(SP),4$  ;;PUT MSG ADDR IN JSR LINKAGE
021466 062766 000002 000004        ADD   #2,4(SP)  ;;BUMP RETURN ADDRESS
021474 013746 177776                MOV   177776,-(SP)  ;;PUSH 177776 ON STACK
021500 004737 022200                JSR   PC,$TYPE  ;;CALL TYPE MACRO
021504 000000        4$:  .WORD  0
021506        5$:
021506 105737 021574        10$: TSTB  $FFLG    ;;SHOULD REPORT FATAL ERROR?
021512 001416                BEQ   12$     ;;IF NOT: BR
021514 005737 020162                TST   $ENV    ;;RUNNING UNDER APT?
021520 001413                BEQ   12$     ;;IF NOT: BR
021522 005737 020142        11$: TST   $MSGTYPE  ;;FINISHED LAST MESSAGE?
021526 001375                BNE   11$     ;;IF NOT: WAIT
021530 017637 000004 020144        MOV   @4(SP),$FATAL  ;;GET ERROR #
021536 062766 000002 000004        ADD   #2,4(SP)  ;;BUMP RETURN ADDR.
021544 005237 020142                INC   $MSGTYPE  ;;TELL APT TO TAKE ERROR
021550 105037 021574        12$: CLRB  $FFLG    ;;CLEAR FATAL FLAG
021554 105037 021573                CLRB  $LFLG    ;;CLEAR LOG FLAG
021560 105037 021572                CLRB  $MFLG    ;;CLEAR MESSAGE FLAG
021564 012601                MOV   (SP)+,R1  ;;POP STACK INTO R1
021566 012600                MOV   (SP)+,R0  ;;POP STACK INTO R0
021570 000207                RTS   PC      ;;RETURN
021572 000                $MFLG: .BYTE  0  ;;MESSG. FLAG
021573 000                $LFLG: .BYTE  0  ;;LOG FLAG
021574 000                $FFLG: .BYTE  0  ;;FATAL FLAG
                .EVEN
000200        APTSIZE=200
000001        APTENV=001
000100        APTSPOOL=100
000040        APTCSUP=040

```

4103

.SBTTL END OF PASS ROUTINE

```

:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
:*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'
:*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
:*IF SW12=1 INHIBIT TRACE TRAP
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO LOOP
    
```

```

021576 000004
021576 005037 177572
021600 005037 172516
021604 104414
021610 005037 001100
021612 005037 001212
021616 005237 020150
021622 042737 100000 020150
021626 005327
021634 000001
021636 003075
021640 012737
021642 000001
021644 021636
021646 104401 021656
021650 000407
021654
021674
021674 013746 020150
021700 104405
021702 005737 001110
021706 001427
021710 104401 021716
021714 000421
021760
021760 013746 001110
021764 104405
021766 104401 001223
021772 005037 001110
021776 013700 000042
022002 001414
022004 005046
022006 012746 022014
022012 000426
022014
022014 013700 000042
022020 001405
022022 000005
022024 004710
022026 000240
022030 000240

$EOP:
SCOPE ;LOOP ON LAST TEST
CLR MMR0 ;TURN OFF FULL RELOCATION
CLR MMR3 ;DISABLE THE UNIBUS MAP
TBITR ;RESTORE THE T BIT IF IT WAS ON
CLR $TSTNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
TYPE ,65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$: MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TST $ERTTL ;SEE IF THERE ARE ANY ERRORS TO REPORT ;DPM001
BEQ 1000$ ;BRANCH AROUND MESSAGE PRINT IF NOT ;DPM001
TYPE ,67$ ;;TYPE ASCIZ STRING
BR 66$ ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$: MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
;;TOTAL NUMBER OF ERRORS
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
1000$: TYPE ,$CRLF ;;TYPE CARRIAGE RETURN, LINE FEED
CLR $ERTTL ;;CLEAR ERROR TOTAL
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
CLR -(SP) ;;INSURE THE 'T' BIT IS CLEAR
MOV #SCLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
BR $RTRN ;;GO DO AN RTI OR RTT TO LOAD THE PSW
;;WITH A CLEARED 'T' BIT

$CLR.T:
MOV @#42,R0 ;;INSURE R0 CONTAINS THE MONITORS
BEQ $DOAGN ;;RETURN ADDRESS
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
    
```


4105

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

.*SAVE R0-R5

.*CALL:

.* SAVREG

.*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:

.*

.*TOP---(+16)

.* +2---(+18)

.* +4---R5

.* +6---R4

.* +8---R3

.*+10---R2

.*+12---R1

.*+14---R0

.\$SAVREG:

022104			MOV	R0,-(SP)	:::PUSH R0 ON STACK
022104	010046		MOV	R1,-(SP)	:::PUSH R1 ON STACK
022106	010146		MOV	R2,-(SP)	:::PUSH R2 ON STACK
022110	010246		MOV	R3,-(SP)	:::PUSH R3 ON STACK
022112	010346		MOV	R4,-(SP)	:::PUSH R4 ON STACK
022114	010446		MOV	R5,-(SP)	:::PUSH R5 ON STACK
022116	010546		MOV	22(SP),-(SP)	:::SAVE PS OF MAIN FLOW
022120	016646	000022	MOV	22(SP),-(SP)	:::SAVE PC OF MAIN FLOW
022124	016646	000022	MOV	22(SP),-(SP)	:::SAVE PS OF CALL
022130	016646	000022	MOV	22(SP),-(SP)	:::SAVE PC OF CALL
022134	016646	000022	MOV	22(SP),-(SP)	:::SAVE PC OF CALL
022140	000002		RTI		

.*RESTORE R0-R5

.*CALL:

.* RESREG

.\$RESREG:

022142			MOV	(SP)+,22(SP)	:::RESTORE PC OF CALL
022142	012666	000022	MOV	(SP)+,22(SP)	:::RESTORE PS OF CALL
022146	012666	000022	MOV	(SP)+,22(SP)	:::RESTORE PC OF MAIN FLOW
022152	012666	000022	MOV	(SP)+,22(SP)	:::RESTORE PS OF MAIN FLOW
022156	012666	000022	MOV	(SP)+,R5	:::POP STACK INTO R5
022162	012605		MOV	(SP)+,R4	:::POP STACK INTO R4
022164	012604		MOV	(SP)+,R3	:::POP STACK INTO R3
022166	012603		MOV	(SP)+,R2	:::POP STACK INTO R2
022170	012602		MOV	(SP)+,R1	:::POP STACK INTO R1
022172	012601		MOV	(SP)+,R0	:::POP STACK INTO R0
022174	012600		MOV	(SP)+,R0	:::POP STACK INTO R0
022176	000002		RTI		

4107

```

.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
:*
$TYPE: TSTB $TFPLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV R0,-(SP) ;;SAVE R0
MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
61$: .WORD 0 ;;MESSAGE ADDRESS
62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
BNE 60$ ;;YES,SKIP TYPE OUT
2$: MOV (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,R0 ;;RESTORE R0
3$: ADD #2,(SP) ;;ADJUST RETURN PC
RTI ;;RETURN
4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
BR 2$ ;;GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;;IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;;GO TYPE A NULL
DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
BR 7$ ;;LOOP
;HORIZONTAL TAB PROCESSOR
8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE

```

```

022372 004737 022412          9$:   JSR   PC,$TYPEC      ;;TYPE A SPACE
022376 132737 000007 022540  BITB  #7,$CHARCNT    ;;BRANCH IF NOT AT
022404 001372          BNE   9$           ;;TAB STOP
022406 005726          TST   (SP)+        ;;POP SPACE OFF STACK
022410 000724          BR    2$           ;;GET NEXT CHARACTER
022412          $TYPEC:
022412 105777 156524          TSTB  @STKS        ;;CHAR IN KYBD BUFFER?
022416 100022          BPL   10$         ;;BR IF NOT
022420 017746 156520          MOV   @STKB,-(SP)  ;;GET CHAR
022424 042716 177600          BIC   #177600,(SP) ;;STRIP EXTRANEIOUS BITS
022430 122716 000023          CMPB  #$XOFF,(SP) ;;WAS CHAR XOFF
022434 001012          BNE   102$       ;;BR IF NOT
022436          101$:
022436 105777 156500          TSTB  @STKS        ;;WAIT FOR CHAR
022442 100375          BPL   101$
022444 117716 156474          MOVB  @STKB,(SP)  ;;GET CHAR
022450 042716 177600          BIC   #177600,(SP) ;;STRIP IT
022454 122716 000021          CMPB  #$XON,(SP)  ;;WAS IT XON?
022460 001366          BNE   101$       ;;BR IF NOT
022462          102$:
022462 005726          TST   (SP)+        ;;FIX STACK
022464          10$:
022464 105777 156456          TSTB  @STPS        ;;WAIT UNTIL PRINTER IS READY
022470 100375          BPL   10$
022472 126627 000002 000021  CMPB  2(SP),#$XON  ;;IS CHARACTER A RANDOM XON?
022500 001420          BEQ   $TYPEX      ;;BRANCH IF YES
022502 116677 000002 156440  MOVB  2(SP),@STPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
022510 122766 000015 000002  CMPB  #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
022516 001003          BNE   1$         ;;BRANCH IF NO
022520 105037 022540          CLRB  $CHARCNT    ;;YES--CLEAR CHARACTER COUNT
022524 000406          BR    $TYPEX     ;;EXIT
022526 122766 000012 000002  1$:  CMPB  #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
022534 001402          BEQ   $TYPEX     ;;BRANCH IF YES
022536 105227          INCB  (PC)+      ;;COUNT THE CHARACTER
022540 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
022542 000207          $TYPEX: RTS    PC

```

4109

```

.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
022544 017646 000000 022767 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
022550 116637 000001 022767 MOV 1(SP), $OFILL ;;LOAD ZERO FILL SWITCH
022556 112637 022771 MOV  (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
022562 062716 000002 ADD #2, (SP) ;;ADJUST RETURN ADDRESS
022566 000406 BR $TYPON
022570 112737 000001 022767 $TYPOC: MOV #1, $OFILL ;;SET THE ZERO FILL SWITCH
022576 112737 000006 022771 MOV #6, $OMODE+1 ;;SET FOR SIX(6) DIGITS
022604 112737 000005 022766 $TYPON: MOV #5, $OCNT ;;SET THE ITERATION COUNT
022612 010346 MOV R3, -(SP) ;;SAVE R3
022614 010446 MOV R4, -(SP) ;;SAVE R4
022616 010546 MOV R5, -(SP) ;;SAVE R5
022620 113704 022771 MOV $OMODE+1, R4 ;;GET THE NUMBER OF DIGITS TO TYPE
022624 005404 NEG R4
022626 062704 000006 ADD #6, R4 ;;SUBTRACT IT FOR MAX. ALLOWED
022632 110437 022770 MOV R4, $OMODE ;;SAVE IT FOR USE
022636 113704 022767 MOV $OFILL, R4 ;;GET THE ZERO FILL SWITCH
022642 016605 000012 MOV 12(SP), R5 ;;PICKUP THE INPUT NUMBER
022646 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
022650 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
022652 000404 BR 3$ ;;GO DO MSB
022654 006105 2$: ROL R5 ;;FORM THIS DIGIT
022656 006105 ROL R5
022660 006105 ROL R5
022662 010503 MOV R5, R3
022664 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
022666 105337 022770 DECB $OMODE ;;TYPE THIS DIGIT?
022672 100016 BPL 7$ ;;BR IF NO
022674 042703 177770 BIC #177770, R3 ;;GET RID OF JUNK
022700 001002 BNE 4$ ;;TEST FOR 0
022702 005704 TST R4 ;;SUPPRESS THIS 0?
022704 001403 BEQ 5$ ;;BR IF YES
022706 005204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
022710 052703 000060 BIS #'0, R3 ;;MAKE THIS DIGIT ASCII
022714 052703 000040 5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY

```


4111

```

.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
$TYPDS:
MOV     R0,-(SP)           ;;PUSH R0 ON STACK
MOV     R1,-(SP)           ;;PUSH R1 ON STACK
MOV     R2,-(SP)           ;;PUSH R2 ON STACK
MOV     R3,-(SP)           ;;PUSH R3 ON STACK
MOV     R5,-(SP)           ;;PUSH R5 ON STACK
MOV     #20200,-(SP)       ;;SET BLANK SWITCH AND SIGN
MOV     20(SP),R5          ;;GET THE INPUT NUMBER
BPL     1$                 ;;BR IF INPUT IS POS.
NEG     R5                 ;;MAKE THE BINARY NUMBER POS.
MOVB   #'-,1(SP)          ;;MAKE THE ASCII NUMBER NEG.
1$:    CLR     R0           ;;ZERO THE CONSTANTS INDEX
MOV     #SDBLK,R3         ;;SETUP THE OUTPUT POINTER
MOVB   #' ,(R3)+          ;;SET THE FIRST CHARACTER TO A BLANK
2$:    CLR     R2           ;;CLEAR THE BCD NUMBER
MOV     $DTBL(R0),R1      ;;GET THE CONSTANT
3$:    SUB     R1,R5        ;;FORM THIS BCD DIGIT
BLT     4$                 ;;BR IF DONE
INC     R2                 ;;INCREASE THE BCD DIGIT BY 1
BR      3$
4$:    ADD     R1,R5        ;;ADD BACK THE CONSTANT
TST     R2                 ;;CHECK IF BCD DIGIT=0
BNE     5$                 ;;FALL THROUGH IF 0
TSTB   (SP)               ;;STILL DOING LEADING 0'S?
BMI     7$                 ;;BR IF YES
5$:    ASLB   (SP)          ;;MSD?
BCC     6$                 ;;BR IF NO
MOVB   1(SP),-1(R3)       ;;YES--SET THE SIGN
6$:    BIS     #'0,R2       ;;MAKE THE BCD DIGIT ASCII
7$:    BIS     #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST     (R0)+             ;;JUST INCREMENTING
CMP     R0,#10           ;;CHECK THE TABLE INDEX
BLT     2$                 ;;GO DO THE NEXT DIGIT
BGT     8$                 ;;GO TO EXIT
MOV     R5,R2             ;;GET THE LSD
BR      6$                 ;;GO CHANGE TO ASCII
8$:    TSTB   (SP)+         ;;WAS THE LSD THE FIRST NON-ZERO?
BPL     9$                 ;;BR IF NO
MOVB   -1(SP),-2(R3)     ;;YES--SET THE SIGN FOR TYPING
9$:    CLRB   (R3)         ;;SET THE TERMINATOR
MOV     (SP)+,R5          ;;POP STACK INTO R5
MOV     (SP)+,R3          ;;POP STACK INTO R3
MOV     (SP)+,R2          ;;POP STACK INTO R2
MOV     (SP)+,R1          ;;POP STACK INTO R1
MOV     (SP)+,R0          ;;POP STACK INTO R0
TYPE   ,SDBLK           ;;NOW TYPE THE NUMBER

```

```

022772
022772 010046
022774 010146
022776 010246
023000 010346
023002 010546
023004 012746 020200
023010 016605 000020
023014 100004
023016 005405
023020 112766 000055 000001
023026 005000 1$:
023030 012703 023206
023034 112723 000040 2$:
023040 005002 023176
023042 016001 3$:
023046 160105
023050 002402
023052 005202
023054 000774
023056 060105 4$:
023060 005702
023062 001002
023064 105716
023066 100407
023070 106316 5$:
023072 103003
023074 116663 000001 177777
023102 052702 000060 6$:
023106 052702 000040 7$:
023112 110223
023114 005720
023116 020027 000010
023122 002746
023124 003002
023126 010502
023130 000764
023132 105726 8$:
023134 100003
023136 116663 177777 177776
023144 105013 9$:
023146 012605
023150 012603
023152 012602
023154 012601
023156 012600
023160 104401 023206

```

```
023164 016666 000002 000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
023172 012616                      MOV      (SP)+,(SP)
023174 000002                      RTI          ;;RETURN TO USER
023176 023420      $DTBL: 10000.
023200 001750                      1000.
023202 000144                      100.
023204 000012                      10.
023206                      $DBLK: .BLKW 4
```

4113

```

.SBTTL TTY INPUT ROUTINE
:*****
:ENABL LSB
:DSABL LSB
:*****
: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
:
$RDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC
023216 011646      000004 000002 1$: MOV      4(SP),2(SP)      ::SAVE THE PS
023220 016666      155710      TSTB     @STKS           ::WAIT FOR
023226 105777      155710      BPL      1$             ::A CHARACTER
023232 100375      155704 000004 2$: MOVB     @STKB,4(SP)      ::READ THE TTY
023234 117766      177600 000004  BIC      #^C<177>,4(SP)  ::GET RID OF JUNK IF ANY
023242 042766      000004 000023  CMP      4(SP),#23      ::IS IT A CONTROL-S?
023250 026627      000004      BNE      3$             ::BRANCH IF NO
023256 001013      155656 2$: TSTB     @STKS           ::WAIT FOR A CHARACTER
023260 105777      155652  BPL      2$             ::LOOP UNTIL ITS THERE
023264 100375      177600  MOVB     @STKB,-(SP)     ::GET CHARACTER
023266 117746      000021  BIC      #^C177,(SP)    ::MAKE IT 7-BIT ASCII
023272 042716      000021  CMP      (SP)+,#21      ::IS IT A CONTROL-Q?
023276 022627      000021  BNE      2$             ::IF NOT DISCARD IT
023302 001366      000004 000021 3$: BR       1$             ::YES, RESUME
023304 000750      000004 000140  CMP      4(SP),#$XON    ::IS IT A RANDOM XON?
023306 026627      000140  BEQ      1$             ::BRANCH IF YES
023314 001744      000004 000175  CMP      4(SP),#140     ::IS IT UPPER CASE?
023316 026627      000004 000175  BLT      4$             ::BRANCH IF YES
023324 002407      000004 000004  CMP      4(SP),#175     ::IS IT A SPECIAL CHAR?
023326 026627      000040 000004  BGT      4$             ::BRANCH IF YES
023334 003003      000040  RTI      #40,4(SP)     ::MAKE IT UPPER CASE
023336 042766      000002  ::GO BACK TO USER
023344 000002
:*****
: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
:
$RDLIN: MOV      R3,-(SP)      ::SAVE R3
023346 010346      023454 1$: MOV      #STTYIN,R3    ::GET ADDRESS
023350 012703      023464 2$: CMP      #STTYIN+8.,R3  ::BUFFER FULL?
023354 022703      101405  BLOS     4$             ::BR IF YES
023360 101405      104406  RDCHR    ::GO READ ONE CHARACTER FROM THE TTY
023362 104406      112613  MOVB     (SP)+,(R3)     ::GET CHARACTER
023364 112613      000177 10$: CMPB    #177,(R3)     ::IS IT A RUBOUT
023366 122713      001222 4$: BNE     3$             ::SKIP IF NOT
023372 001003      023452 3$: TYPE    ,SQUES       ::TYPE A '?'
023374 104401      023452  BR      1$             ::CLEAR THE BUFFER AND LOOP
023400 000763      023452  MOVB     (R3),9$       ::ECHO THE CHARACTER
023402 111337      000015  TYPE    ,9$
023406 104401      000015  CMPB    #15,(R3)+     ::CHECK FOR RETURN
023412 122723      000015  BNE     2$             ::LOOP IF NOT RETURN
023416 001356      177777  CLRB    -1(R3)        ::CLEAR RETURN (THE 15)
023420 105063      001224  TYPE    ,LF           ::TYPE A LINE FEED
023424 104401      012603  MOV     (SP)+,R3      ::RESTORE R3
023430 012603

```

023432	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
023434	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
023442	012766	023454	000004	MOV	#\$TTYIN,4(SP)	
023450	000002			RTI		::RETURN
023452	000			9\$: .BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
023453	000			.BYTE	0	::TERMINATOR
023454				\$TTYIN: .BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
023464	136	125	015	\$CNTLU: .ASCIZ	/^U/<15><12>	::CONTROL 'U'
023471	136	107	015	\$CNTLG: .ASCIZ	/^G/<15><12>	::CONTROL 'G'
023476	015	012	123	\$MSWR: .ASCIZ	<15><12>/SWR = /	
023507	040	040	116	\$MNEW: .ASCIZ	/ NEW = /	

4115

.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.

023520 010046
 023522 016600 000002
 023526 005740
 023530 111000
 023532 006300
 023534 016000 023554
 023540 000200

```
$TRAP:  MOV    RO,-(SP)      ;;SAVE R0
        MOV    2(SP),RO    ;;GET TRAP ADDRESS
        TST   -(RO)       ;;BACKUP BY 2
        MOVB  (RO),RO     ;;GET RIGHT BYTE OF TRAP
        ASL   RO          ;;POSITION FOR INDEXING
        MOV   $TRPAD(RO),RO ;;INDEX TO TABLE
        RTS   RO          ;;GO TO ROUTINE
```

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

023542 011646
 023544 016666 000004 000002
 023552 000002

```
$TRAP2: MOV   (SP),-(SP)    ;;MOVE THE PC DOWN
        MOV   4(SP),2(SP)  ;;MOVE THE PSW DOWN
        RTI                    ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 *BY THE "TRAP" INSTRUCTION.
 ROUTINE

023554 023542
 023556 022200
 023560 022570
 023562 022544
 023564 022604
 023566 022772
 023570 023216
 023572 023346
 023574 005630
 023576 022104
 023600 022142
 4116 023602 002726
 4117 023604 002754
 4118 023606 005630

```
$TRPAD: .WORD  $TRAP2
        $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPOC ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
        $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
        $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
        $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
        $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
        $TBITOF ;;CALL=TBITOF   TRAP+13(104413) THIS WILL TURN OFF T BIT TRAPPING
        $TBITRE ;;CALL=TBITRE   TRAP+14(104414) THIS WILL RETURN THE T BIT TO PREVIOUS CONDI
        $RDOCT  ;;CALL=RDOCT    TRAP+15(104415) READ OCTAL NUMBER
```

4120

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

023610	012737	023766	000024	\$PWRDN: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST UP
023616	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
023624	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
023626	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
023630	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
023632	010346			MOV	R3,-(SP)	::PUSH R3 ON STACK
023634	010446			MOV	R4,-(SP)	::PUSH R4 ON STACK
023636	010546			MOV	R5,-(SP)	::PUSH R5 ON STACK
023640	017746	155272		MOV	@SWR,-(SP)	::PUSH @SWR ON STACK
023644	010637	023772		MOV	SP,\$SAVR6	::SAVE SP
023650	012737	023662	000024	MOV	#\$PWRUP,@#PWRVEC	::SET UP VECTOR
023656	000000			HALT		
023660	000776			BR	.-2	::HANG UP

:POWER UP ROUTINE

023662	012737	023766	000024	\$PWRUP: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST DOWN
023670	013706	023772		MOV	\$SAVR6,SP	::GET SP
023674	005037	023772		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
023700	005237	023772		1\$: INC	\$SAVR6	::WAIT FOR THE INC
023704	001375			BNE	1\$::OF WORD
023706	012677	155224		MOV	(SP)+,@SWR	::POP STACK INTO @SWR
023712	012605			MOV	(SP)+,R5	::POP STACK INTO R5
023714	012604			MOV	(SP)+,R4	::POP STACK INTO R4
023716	012603			MOV	(SP)+,R3	::POP STACK INTO R3
023720	012602			MOV	(SP)+,R2	::POP STACK INTO R2
023722	012601			MOV	(SP)+,R1	::POP STACK INTO R1
023724	012600			MOV	(SP)+,R0	::POP STACK INTO R0
023726	012737	023610	000024	MOV	#\$PWRDN,@#PWRVEC	::SET UP THE POWER DOWN VECTOR
023734	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
023742	104401			TYPE		::REPORT THE POWER FAILURE
023744	023774			SPWRMG: .WORD	PWRMSG	::POWER FAIL MESSAGE POINTER
023746	012716			MOV	(PC)+,(SP)	::RESTART AT START
023750	010000			SPWRAD: .WORD	START	::RESTART ADDRESS
023752	042766	000020	000002	BIC	#20,2(SP)	::CLEAR 'T' BIT
023760	005037	022076		CLR	\$TBIT	::CLEAR THE 'T' BIT FLAG
023764	000002			RTI		
023766	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
023770	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
023772	000000			\$SAVR6: 0		::PUT THE SP HERE
4121	023774	015	120	PWRMSG: .ASCIZ	<12><15>?POWER FAILURE, RESTARTING PROGRAM?	
4122					.EVEN	

4124

```

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
:*****
:*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
:*UNSIGNED OCTAL ASCII NUMBER.
:*CALL
:*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
:*      JSR      PC,@#$DB20      ;; CALL THE ROUTINE
:*      RETURN   ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
$DB20: SAVREG      ;; SAVE ALL REGISTERS
      MOV      2(SP),R1          ;; PICKUP THE POINTER TO LOW WORD
      MOV      #$OCTVL+13.,R5    ;; POINTER TO DATA TABLE
      MOV      #12.,R4           ;; DO ELEVEN CHARACTERS
      MOV      #^C7,R3           ;; MASK
      MOV      (R1)+,R0          ;; LOWER WORD
      MOV      (R1)+,R1          ;; HIGH WORD
      CLR      R2                ;; TERMINATOR
1$:    MOVB     R2,-(R5)          ;; PUT CHARACTER IN DATA TABLE
      MOV      R0,R2            ;; GET THIS DIGIT
      DEC      R4               ;; COUNT THIS CHARACTER
      BGT      3$              ;; BR IF NOT THE LAST DIGIT
      BEQ      2$              ;; BR IF IT IS THE LAST DIGIT
      INC      R5               ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
      MOV      R5,2(SP)         ;; ASCII CHAR. & PUT IT ON THE STACK
      RESREG   ;; RESTORE ALL REGISTERS
      RTS      PC              ;; RETURN TO USER
2$:    ASR      R3              ;; POSITION THE MASK FOR THE LAST DIGIT
3$:    ROR      R1              ;; POSITION THE BINARY NUMBER FOR
      ROR      R0              ;; THE NEXT OCTAL DIGIT
      ROR      R1
      ROR      R0
      ROR      R1
      ROR      R0
      BIC      R3,R2           ;; MASK OUT ALL JUNK
      ADD      #'0,R2         ;; MAKE THIS CHAR. ASCII
      BR       1$             ;; GO PUT IT IN THE DATA TABLE
$OCTVL: .BLKB 14.           ;; RESERVE DATA TABLE

```

```

024040 104411
024042 016601 000002
024046 012705 024157
024052 012704 000014
024056 012703 177770
024062 012100
024064 012101
024066 005002
024070 110245
024072 010002
024074 005304
024076 003007
024100 001405
024102 005205
024104 010566 000002
024110 104412
024112 000207
024114 006203
024116 006001
024120 006000
024122 006001
024124 006000
024126 006001
024130 006000
024132 040302
024134 062702 000060
024140 000753
024142

```

Line	Address	Line	Line	Line	SBTTL	ERROR MESSAGES
4126						
4127	024160	116	117	124	EM1:	.ASCIZ ?NOT THE CORRECT TRAP CONDITION THROUGH ERRVEC (#004)?
4128	024245	125	116	105	EM2:	.ASCIZ ?UNEXPECTED CPU TRAP THROUGH ERRVEC (#004)?
4129	024317	125	116	105	EM3:	.ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?
4130	024425	123	125	115	EM4:	.ASCIZ ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?
4131	024505	123	125	115	EM5:	.ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS?
4132	024600	123	125	115	EM6:	.ASCIZ ?SUMMARY OF BIT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS?
4133	024702	123	125	115	EM7:	.ASCIZ ?SUMMARY OF BIT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS?
4134	025003	103	101	116	EM10:	.ASCII ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF>
4135	025071	123	117	040		.ASCIZ ?SO JUMPING TO THE SIZE JUMPER TEST FOR VERIFICATION?
4136	025155	123	125	115	EM11:	.ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?
4137	025247	125	116	111	EM12:	.ASCIZ ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?
4138	025321	103	101	116	EM13:	.ASCII ?CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL?<CRLF>
4139	025401	101	104	104		.ASCII ?ADDRESS BIT14 IS STUCK LOW, MUST RESTART PROGRAM?<CRLF>
4140	025462	111	106	040		.ASCIZ ?IF YOU DON'T LOOP ON THIS PROBLEM.?
4141	025525	123	125	115	EM14:	.ASCIZ ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED?
4142	025630	115	101	111	EM15:	.ASCIZ ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?
4143	025723	122	105	114	EM16:	.ASCIZ ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?
4144	026021	116	117	040	EM17:	.ASCIZ ?NO UNIBUS MEMORY EXISTS?
4145	026051	111	116	124	EM20:	.ASCIZ ?INTERRUPT/ABORT LOGIC TESTS TRAP TO LOCATION 114 DID NOT OCCUR?
4146	026150	111	116	124	EM21:	.ASCIZ ?INTERRUPT/ABORT TESTS R4 WAS OVERWRITTEN WITH?
4147	026226	111	116	124	EM22:	.ASCIZ ?INTERRUPT/ABORT TESTS TRAP DID NOT OCCUR DUE TO ABORT?
4148	026314	114	115	101	EM23:	.ASCIZ ?LMA NOT LOADED PROPERLY?
4149	026344	114	115	101	EM24:	.ASCIZ ?LMA FORCE JUMPER BIT NOT ZERO?
4150	026402	114	115	101	EM25:	.ASCIZ ?LMA FORCE JUMPER BIT NOT SET?
4151	026437	114	115	101	EM26:	.ASCIZ ?LMA CONTROL BITS INCORRECT?
4152	026472	106	117	122	EM27:	.ASCIZ ?FORCE JUMPER BIT FAILS TO REVERT MAP REGISTER STATUS TO DEFAULT?
4153	026572	113	111	120	EM30:	.ASCIZ ?KIPAR5 NOT LOADED PROPERLY?
4154	026625	124	110	105	EM201:	.ASCIZ ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ?
4155	026701	124	110	105	EM202:	.ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP?
4156	026774	124	110	105	EM203:	.ASCIZ ?THE BIT PATTERN THROUGH THE MAP REGISTERS FAILED?
4157	027055	125	116	111	EM204:	.ASCIZ ?UNIBUS DATA PATH COUNT PATTERN FAILURE?
4158	027124	125	116	111	EM205:	.ASCIZ ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED?
4159	027206	104	101	124	EM206:	.ASCIZ ?DATA PATTERN NOT CORRECT?
4160	027237	122	105	106	EM207:	.ASCIZ ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 17770200?
4161	027346	115	101	120	EM210:	.ASCIZ ?MAP REGISTER(S) UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?
4162	027446	122	105	114	EM211:	.ASCII ?RELOCATION THROUGH THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>
4163	027544	124	105	123		.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?
4164	027604	115	101	111	EM212:	.ASCII ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CRLF>
4165	027677	124	105	123		.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?
4166	027737	115	101	120	EM213:	.ASCIZ ?MAP REGISTER ENABLED WHEN DDW SAYS IT SHOULD BE DISABLED?
4167	030030	115	101	120	EM214:	.ASCIZ ?MAP REGISTER DISABLED WHEN DDW SAYS IT SHOULD BE ENABLED?

Line	Code	Line	Code	Line	Code	Line	Code	Line	Code	Line	Code	Line	Code
4168													
4169	030121	122	105	103	DH1:	.ASCIZ	?RECEIVD	EXPECTD	TESTNO	ERR	PC?		
4170	030160	122	105	103	DH2:	.ASCIZ	?RECEIVD	TESTNO	ERR	PC?			
4171	030207	123	124	101	DH3:	.ASCII	?STATUS	AUTOI/D	VIRTUAL?	<CRLF>			
4172	030237	122	105	107		.ASCIZ	?REGISTR	REGISTR	ADDRESS	TESTNO	ERR	PC?	
4173	030306	122	105	107	DH4:	.ASCII	?REGADRS	REGADRS?	<CRLF>				
4174	030330	040	042	117		.ASCIZ	? 'OR'	'AND'	#ERRORS	TESTNO	ERR	PC?	
4175	030403	122	105	107	DH5:	.ASCII	?REGLOAD	REGLOAD	REGDUAL	REGDUAL?	<CRLF>		
4176	030451	040	042	117		.ASCIZ	? 'OR'	'AND'	'OR'	'AND'	#ERRORS	TESTNO?	
4177	030540	115	101	120	DH6:	.ASCII	?MAPREG	MAPREG	EXPECTD	EXPECTD	RECEIVD	RECEIVD?	<CRLF>
4178	030624	040	042	117		.ASCIZ	? 'OR'	'AND'	'OR'	'AND'	'OR'	'AND'	#ERRORS
4179	030727	124	105	123	DH10:	.ASCIZ	?TESTNO	ERR	PC	MMR3?			
4180	030754	105	130	120	DH11:	.ASCII	?EXPECTD	EXPECTD	RECEIVD	RECEIVD?	<CRLF>		
4181	031022	040	042	117		.ASCIZ	? 'OR'	'AND'	'OR'	'AND'	#ERRORS	TESTNO?	
4182	031111	103	117	116	DH15:	.ASCII	?CONDITN	CONDITN?	<CRLF>				
4183	031131	105	130	120		.ASCIZ	?EXPECTD	RECEIVD	TESTNO	ERR	PC?		
4184	031170	124	105	123	DH23:	.ASCIZ	?TESTNO	ERR	PC	LMAEXP	LMARCV?		
4185	031231	124	105	123	DH24:	.ASCIZ	?TESTNO	ERR	PC	LMAEXP	LMARCV?		
4186	031270	124	105	123	DH27:	.ASCIZ	?TESTNO	ERR	PC	LMARCV	KIPAR4?		
4187	031327	124	105	123	DH30:	.ASCIZ	?TESTNO	ERR	PC	PR5EXP	PR5RCV?		
4188	031366	122	105	107	DH201:	.ASCIZ	?REGADRS	TESTNO	ERR	PC?			
4189	031415	115	101	120	DH202:	.ASCII	?MAPREG	MAPREG	NON-ZER?	<CRLF>			
4190	031451	124	105	123		.ASCIZ	?TESTING	DUALED	CONTNTS	TESTNO	ERR	PC?	
4191	031524	122	105	107	DH203:	.ASCIZ	?REGADRS	PATRN	EXPCTD	RECEVD	TESTNO	ERR	PC?
4192	031605	105	130	120	DH204:	.ASCIZ	?EXPECTD	RECEIVD	ADDRSLOAD	TESTNO	ERR	PC?	
4193	031656	101	104	104	DH205:	.ASCII	?ADDRESS	ADDRESS?	<CRLF>				
4194	031700	105	130	120		.ASCIZ	?EXPECTD	RECEIVD	TESTNO	ERR	PC?		
4195	031743	101	104	104	DH206:	.ASCIZ	?ADDRESS	EXPCTD	RECVD	TESTNO	ERR	PC?	
4196	032013	101	104	104	DH207:	.ASCIZ	?ADDRUSED	BITDIFF	TESTNO	ERR	PC?		
4197	032054	124	105	123	DH210:	.ASCIZ	?TESTNO	ERR	PC	MAPREGADR?			
4198	032106	103	117	122	DH211:	.ASCII	?CORRECT	EXPECTD	RECEIVD?	<CRLF>			
4199	032140	101	104	104		.ASCIZ	?ADDRESS	DATA	FROM	UB	TESTNO	ERR	PC?
4200	032211	103	117	116	DH212:	.ASCII	?CONDITN	CONDITN?	<CRLF>				
4201	032231	105	130	120		.ASCIZ	?EXPECTD	RECEIVD	TESTNO	ERR	PC?		
4202	032270	124	105	123	DH213:	.ASCIZ	?TESTNO	ERR	PC	REG	NO	DDWDAT	DDWADR?
4203						.EVEN							

					.SBTTL	DATA TABLES
4204						
4205	032340	001330	001326	020146	DT1:	.WORD PCPUER,CPUEXP,\$TESTN,BADPC,0
4206	032352	001330	020146	001340	DT2:	.WORD PCPUER,\$TESTN,BADPC,0
4207	032362	001350	001352	001354	DT3:	.WORD PMMR0,PMMR1,PMMR2,\$TESTN,BADPC,0
4208	032376	001236	001232	001320	DT4:	.WORD ADDROR,ADRAND,ERRCNT,\$TESTN,\$ERRPC,0
4209	032412	001236	001232	001246	DT5:	.WORD ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
4210	032430	001236	001232	001254	DT6:	.WORD ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
4211	032452	020146	001114	172516	DT10:	.WORD \$TESTN,\$ERRPC,MMR3,0
4212	032462	001254	001252	001246	DT11:	.WORD PATTOR,PATAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
4213	032500	001236	001232	001246	DT14:	.WORD ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,\$TESTN,0
4214	032516	001326	001330	020146	DT15:	.WORD CPUEXP,PCPUER,\$TESTN,\$ERRPC,0
4215	032530	020146	001114	005764	DT23:	.WORD \$TESTN,\$ERRPC,EADRES,EADRS2,0
4216	032542	020146	001114	001162	DT24:	.WORD \$TESTN,\$ERRPC,\$REG1,LMAHI,0
4217	032554	020146	001114	001174	DT26:	.WORD \$TESTN,\$ERRPC,\$TMP0,\$REG2,0
4218	032566	020146	001114	001174	DT27:	.WORD \$TESTN,\$ERRPC,\$TMP0,KIPAR4,0
4219	032600	020146	001114	001206	DT30:	.WORD \$TESTN,\$ERRPC,\$TMP5,KIPAR5,0
4220	032612	005764	020146	001114	DT201:	.WORD EADRES,\$TESTN,\$ERRPC,0
4221	032622	005770	005764	001202	DT202:	.WORD EADRS2,EADRES,\$TMP3,\$TESTN,\$ERRPC,0
4222	032636	005770	001174	001170	DT203:	.WORD EADRS2,\$TMP0,\$REG4,\$REG3,\$TESTN,\$ERRPC,0
4223	032654	001174	001176	001164	DT204:	.WORD \$TMP0,\$TMP1,\$REG2,\$TESTN,\$ERRPC,0
4224	032670	005764	005770	020146	DT205:	.WORD EADRES,EADRS2,\$TESTN,\$ERRPC,0
4225	032702	005764	001204	001206	DT206:	.WORD EADRES,\$TMP4,\$TMP5,\$TESTN,\$ERRPC,0
4226	032716	005764	001160	020146	DT207:	.WORD EADRES,\$REG0,\$TESTN,\$ERRPC,0
4227	032730	020146	001114	005764	DT210:	.WORD \$TESTN,\$ERRPC,EADRES,0
4228	032740	005764	001166	001164	DT211:	.WORD EADRES,\$REG3,\$REG2,\$TESTN,\$ERRPC,0
4229	032754	001326	001330	020146	DT212:	.WORD CPUEXP,PCPUER,\$TESTN,\$ERRPC,0
4230	032766	020146	001114	001174	DT213:	.WORD \$TESTN,\$ERRPC,\$TMP0,\$TMP1,\$REG5,0

4231					.SBTTL	NON-DEFAULT UNIBUS MAP JUMPER MESSAGE
4232	033002	200	123	111	JMPMSG: .ASCII	<CRLF>?SIZE JUMPERS ON UNIBUS MAP ARE NOT IN THEIR DEFAULT?<CRLF>
4233	033067	120	117	123	.ASCII	?POSITION. MAP REGISTERS BETWEEN THE LOWEST AND HIGHEST?<CRLF>
4234	033157	125	123	105	.ASCII	?USEABLE, AND ABOVE THE UNIBUS END NUMBER WILL BE TESTED.?<CRLF>
4235	033250	125	116	111	.ASCII	?UNIBUS MEMORY WILL BE ASSUMED TO BE BETWEEN UNIBUS BEGIN?<CRLF>
4236	033341	101	116	104	.ASCII	?AND END REGISTER NUMBERS IF BIT <5> IS SET IN THE SWR,?<CRLF>
4237	033430	105	116	101	.ASCII	?ENABLING TEST #23 TO EXECUTE.?<CRLF><CRLF>
4238	033467	040	040	114	.ASCII	? LOWEST HIGEST UNIBUS UNIBUS?<CRLF>
4239	033530	040	040	125	.ASCII	? USABLE USABLE BEGIN END?<CRLF>
4240	033567	040	040	040	.ASCIZ	? REG# REG # REG # REG # TEST #?<CRLF>

				.SBTTL	DATA FIELDS	
4241						
4242	033641	000	000	000	DF1:	.BYTE 0,0,0,0,0
4243	033646	002	002	001	DF4:	.BYTE 2,2,1,0,0
4244	033653	002	002	002	DF5:	.BYTE 2,2,2,2,1,0
4245	033661	002	002	000	DF6:	.BYTE 2,2,0,0,0,0,1,0
4246	033671	000	000	000	DF11:	.BYTE 0,0,0,0,1,0
4247	033677	002	002	000	DF14:	.BYTE 2,2,0,0,1,0
4248	033705	000	000	002	DF23:	.BYTE 0,0,2,2
4249	033711	002	000	000	DF201:	.BYTE 2,0,0,0,0,0
4250	033717	000	000	003	DF204:	.BYTE 0,0,3,0,0
4251	033724	000	000	002	DF210:	.BYTE 0,0,2
4252		000001				.END

ABASE = 000000
ACDW1 = 000000
ACDW2 = 000000
ACPUOP = 000000
ADDROR = 001236
ADDW0 = 177777
ADDW1 = 177777
ADDW10 = 000000
ADDW11 = 000000
ADDW12 = 000000
ADDW13 = 000000
ADDW14 = 000000
ADDW15 = 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT = 000000
ADEVM = 000000
ADRAND = 001232
ADREXT = 003674
AENV = 000000
AENVM = 000000
AFATAL = 000000
AMADR1 = 000000
AMADR2 = 000000
AMADR3 = 000000
AMADR4 = 000000
AMAMS1 = 000000
AMAMS2 = 000000
AMAMS3 = 000000
AMAMS4 = 000000
AMSGAD = 000000
AMSGLG = 000000
AMSGTY = 000000
AMTYP1 = 000000
AMTYP2 = 000000
AMTYP3 = 000000
AMTYP4 = 000000
APASS = 000000
APRIOR = 000000
APTCSU = 000040
APTENV = 000001
APTSIZ = 000200
APTSPO = 000100
ASWREG = 000000
ATESTN = 000000
AUNIT = 000000
AUSWR = 000000
AVECT1 = 000000
AVECT2 = 000000
BADCPU = 006607
BADPC = 001340

BIT0 = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1 = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2 = 000004
BIT3 = 000010
BIT4 = 000020
BIT5 = 000040
BIT6 = 000100
BIT7 = 000200
BIT8 = 000400
BIT9 = 001000
BPTVEC = 000014
BUPWIN = 001276
CACHE = 177746
CACHVE = 000114
CASHSR = 005324
CASH1 = 005460
CASH2 = 005470
CHARCT = 005760
CHKLMA = 005076
CHKPAT = 005600
CLRMAT = 002772
CMPE = 177744
CNTR = 001322
CONTRL = 177746
CPFLAG = 003224
CPSAVE = 020700
CPUER = 003222
CPUERR = 177766
CPUEXP = 001326
CPUMSG = 006700
CPUTYP = 007140
CR = 000015
CRLF = 000200
CTRAPS = 000116
CTRAPV = 000114
DATA = 001364
DATAND = 001242
DATAOR = 001246
DATEXT = 003624
DATO = 100000
DATOB = 140000

DDISP = 177570
DFMSG = 006120
DF1 = 033641
DF11 = 033671
DF14 = 033677
DF201 = 033711
DF204 = 033717
DF210 = 033724
DF23 = 033705
DF4 = 033646
DF5 = 033653
DF6 = 033661
DH1 = 030121
DH10 = 030727
DH11 = 030754
DH15 = 031111
DH2 = 030160
DH201 = 031366
DH202 = 031415
DH203 = 031524
DH204 = 031605
DH205 = 031656
DH206 = 031743
DH207 = 032013
DH210 = 032054
DH211 = 032106
DH212 = 032211
DH213 = 032270
DH23 = 031170
DH24 = 031231
DH27 = 031270
DH3 = 030207
DH30 = 031327
DH4 = 030306
DH5 = 030403
DH6 = 030540
DISPLA = 001140
DISPRE = 000174
DSABLD = 005230
DSWR = 177570
DTMS = 005740
DTMSG = 005744
DT1 = 032340
DT10 = 032452
DT11 = 032462
DT14 = 032500
DT15 = 032516
DT2 = 032352
DT201 = 032612
DT202 = 032622
DT203 = 032636
DT204 = 032654
DT205 = 032670
DT206 = 032702
DT207 = 032716
DT210 = 032730
DT211 = 032740

DT212 = 032754
DT213 = 032766
DT23 = 032530
DT24 = 032542
DT26 = 032554
DT27 = 032566
DT3 = 032362
DT30 = 032600
DT4 = 032376
DT5 = 032412
DT6 = 032430
EADRES = 005764
EADRS2 = 005770
EMTVEC = 000030
EM1 = 024160
EM10 = 025003
EM11 = 025155
EM12 = 025247
EM13 = 025321
EM14 = 025525
EM15 = 025630
EM16 = 025723
EM17 = 026021
EM2 = 024245
EM20 = 026051
EM201 = 026625
EM202 = 026701
EM203 = 026774
EM204 = 027055
EM205 = 027124
EM206 = 027206
EM207 = 027237
EM21 = 026150
EM210 = 027346
EM211 = 027446
EM212 = 027604
EM213 = 027737
EM214 = 030030
EM22 = 026226
EM23 = 026314
EM24 = 026344
EM25 = 026402
EM26 = 026437
EM27 = 026472
EM3 = 024317
EM30 = 026572
EM4 = 024425
EM5 = 024505
EM6 = 024600
EM7 = 024702
ENABLD = 005266
ERRCNT = 001320
ERROR = 104000
ERRVEC = 000004
ERTYPE = 002026
ER200 = 001666
EXTOUT = 005762

FJBIT = 000100
FLAG = 001324
FLOATR = 005774
FTTHRU = 003532
GMRMD1 = 007043
GMRMOD = 007046
HIADRS = 177742
HIGEST = 001260
HITMIS = 177752
HT = 000011
IBSAVE = 020702
IOTVEC = 000020
JMPMSG = 033002
KDPAR0 = 172360
KDPAR1 = 172362
KDPAR2 = 172364
KDPAR3 = 172366
KDPAR4 = 172370
KDPAR5 = 172372
KDPAR6 = 172374
KDPAR7 = 172376
KDPDR0 = 172320
KDPDR1 = 172322
KDPDR2 = 172324
KDPDR3 = 172326
KDPDR4 = 172330
KDPDR5 = 172332
KDPDR6 = 172334
KDPDR7 = 172336
KERSTK = 001100
KIPAR0 = 172340
KIPAR1 = 172342
KIPAR2 = 172344
KIPAR3 = 172346
KIPAR4 = 172350
KIPAR5 = 172352
KIPAR6 = 172354
KIPAR7 = 172356
KIPDR0 = 172300
KIPDR1 = 172302
KIPDR2 = 172304
KIPDR3 = 172306
KIPDR4 = 172310
KIPDR5 = 172312
KIPDR6 = 172314
KIPDR7 = 172316
KSP = 000006
LF = 000012
LKS = 177546
LKVEC = 000100
LMAH = 001304
LMAHI = 177736
LMAL = 001306
LMALOW = 177734
LOADRS = 177740
LOEFLG = 004312
LOOP = 010606

LOWEST	001256	MAPL12=	170250	PATEXT	003650	SDPAR7=	172276	SW13	=	020000
LREGL	001300	MAPL13=	170254	PATRNS	006102	SDPDR0=	172220	SW14	=	040000
LREGU	001302	MAPL14=	170260	PATTOR	001254	SDPDR1=	172222	SW15	=	100000
MAINT	=	MAPL15=	170264	PCONTR	001334	SDPDR2=	172224	SW2	=	000004
MAPADD	004106	MAPL16=	170270	PCPUER	001330	SDPDR3=	172226	SW3	=	000010
MAPHO	=	MAPL17=	170274	PIRQ	=	SDPDR4=	172230	SW4	=	000020
MAPHO0=	170202	MAPL2	=	PIRQVE=	000240	SDPDR5=	172232	SW5	=	000040
MAPHO1=	170206	MAPL20=	170300	PMAINT	001336	SDPDR6=	172234	SW6	=	000100
MAPHO2=	170212	MAPL21=	170304	PMBECD	002526	SDPDR7=	172236	SW7	=	000200
MAPHO3=	170216	MAPL22=	170310	PMBECF	002536	SIPAR0=	172240	SW8	=	000400
MAPHO4=	170222	MAPL23=	170314	PMBECH	002476	SIPAR1=	172242	SW9	=	001000
MAPHO5=	170226	MAPL24=	170320	PMBECM	002442	SIPAR2=	172244	SYSTID=	177764	
MAPHO6=	170232	MAPL25=	170324	PMBECW	002432	SIPAR3=	172246	TBIT	=	000020
MAPHO7=	170236	MAPL26=	170330	PMMR0	001350	SIPAR4=	172250	TBITO	=	104413
MAPH1	=	MAPL27=	170334	PMMR1	001352	SIPAR5=	172252	TBITOF	=	002726
MAPH10=	170242	MAPL3	=	PMMR2	001354	SIPAR6=	172254	TBITR	=	104414
MAPH11=	170246	MAPL30=	170340	PPARER	001332	SIPAR7=	172256	TBITRE	=	002754
MAPH12=	170252	MAPL31=	170344	PRETST	005166	SIPDR0=	172200	TBITVE=	000014	
MAPH13=	170256	MAPL32=	170350	PR0	=	SIPDR1=	172202	TCPMRA	004416	
MAPH14=	170262	MAPL33=	170354	PR1	=	SIPDR2=	172204	TIMEOU	003030	
MAPH15=	170266	MAPL34=	170360	PR2	=	SIPDR3=	172206	TIMOUT=	000020	
MAPH16=	170272	MAPL35=	170364	PR3	=	SIPDR4=	172210	TKVEC	=	000060
MAPH17=	170276	MAPL36=	170370	PR4	=	SIPDR5=	172212	TOFLAG	003032	
MAPH2	=	MAPL37=	170374	PR5	=	SIPDR6=	172214	TOMSG	006261	
MAPH20=	170302	MAPL4	=	PR6	=	SIPDR7=	172216	TPVEC	=	000064
MAPH21=	170306	MAPL5	=	PR7	=	SIZEHI=	177762	TRAPVE=	000034	
MAPH22=	170312	MAPL6	=	PS	=	SIZEJ0	012744	TRTVEC=	000014	
MAPH23=	170316	MAPL7	=	PSW	=	SIZEJ1	013154	TSTLOC	003474	
MAPH24=	170320	MASK1	005776	PTMP2	005000	SIZEJ2	013402	TST1	010676	
MAPH25=	170326	MASK2	006000	PWRMSG	023774	SIZELO=	177760	TST10	014616	
MAPH26=	170332	MEMERR=	177744	PWRVEC=	000024	SPECST	006002	TST11	015072	
MAPH27=	170336	MFPT	=	RDCHR	=	SR0	=	TST12	015140	
MAPH3	=	MMFLAG	003374	RDLIN	=	SR1	=	TST13	015174	
MAPH30=	170342	MMRHI	001270	RDOCT	=	SR2	=	TST14	015334	
MAPH31=	170346	MMRLOW	001266	RELC22	011672	SR3	=	TST15	015362	
MAPH32=	170352	MMR0	=	RESREG=	104412	SSP	=	TST16	015420	
MAPH33=	170356	MMR1	=	RESVEC=	000010	STACK	=	TST17	015450	
MAPH34=	170362	MMR2	=	RETRY	001360	START	010000	TST2	011122	
MAPH35=	170366	MMR3	=	RSIZE	001356	STKLMT=	177774	TST20	015610	
MAPH36=	170372	MMTOTM	020014	R10	=	SUPSTK=	000700	TST21	015744	
MAPH37=	170376	MMTRAP	003372	R11	=	SWR	001136	TST22	016220	
MAPH4	=	MMVEC	=	R12	=	SWREG	000176	TST23	016266	
MAPH5	=	MRQUES	006754	R13	=	SW0	=	TST24	016474	
MAPH6	=	NEWSWR	006161	R14	=	SW00	=	TST25	016540	
MAPH7	=	NEXMEM=	000040	R15	=	SW01	=	TST26	016702	
MAPL0	=	NEXT	003552	R3STAK	004314	SW02	=	TST27	017036	
MAPL00=	170200	NOMORE	006421	R6	=	SW03	=	TST3	011444	
MAPL01=	170204	NOMSG	013646	R7	=	SW04	=	TST30	017164	
MAPL02=	170210	NUMOFK	001316	SAVREG=	104411	SW05	=	TST31	017412	
MAPL03=	170214	NXTTST	001362	SCOPE	=	SW06	=	TST32	=	
MAPL04=	170220	OLDPC	001342	SDPAR0=	172260	SW07	=	TST4	011762	
MAPL05=	170224	OLDPS	001344	SDPAR1=	172262	SW08	=	TST5	012242	
MAPL06=	170230	OLDPSW	001346	SDPAR2=	172264	SW09	=	TST6	012630	
MAPL07=	170234	PADRSR	001230	SDPAR3=	172266	SW1	=	TST7	013712	
MAPL1	=	PADRSL	001226	SDPAR4=	172270	SW10	=	TYPDAT=	002244	
MAPL10=	170240	PATAND	001252	SDPAR5=	172272	SW11	=	TYPDS	=	
MAPL11=	170244	PATCH	007142	SDPAR6=	172274	SW12	=	TYPE	=	

TYPOC = 104402	USESTK= 000600	SDOAGN 022034	SMBADR 020002	SSVPC = 000204
TYPON = 104404	USP = %000006	\$DTBL 023176	\$MFLG 021572	\$SWR = 177400
TYPOS = 104403	YESMSG 013542	\$ENDAD 022024	\$MNEW 023507	\$SWREG 020164
TYPVAD 002542	\$APTHD 020000	\$ENDCT 021644	\$MSGAD 020156	\$SWRMK= 000340
UBADDR 002650	\$ATYC 021354	\$ENULL 022100	\$MSGLG 020160	\$TBIT 022076
UBMAVA 006125	\$ATY1 021330	\$ENV 020162	\$MSGTY 020142	\$TESTN 020146
UBMEND 006174	\$ATY3 021336	\$ENVM 020163	\$MSWR 023476	\$TIMES 001212
UBMHI 001264	\$ATY4 021346	\$EOP 021576	\$MTYP1 020173	\$TKB 001144
UBMLOW 001262	\$AUTOB 001132	\$EOPCT 021636	\$MTYP2 020177	\$TKS 001142
UBMSU 015236	\$BASE 020216	\$ERFLG 001101	\$MTYP3 020203	\$TMP0 001174
UBM24L 001310	\$BDADR 001120	\$ERMAX 001113	\$MTYP4 020207	\$TMP1 001176
UBM24P 001314	\$BDDAT 001124	\$ERROR 020704	\$MXCNT 020676	\$TMP2 001200
UBM24U 001312	\$BELL 001216	\$ERRPC 001114	\$NULL 001152	\$TMP3 001202
UBRHI 001274	\$CDW1 020222	\$ERRTB 001366	\$NWTST= 000001	\$TMP4 001204
UBRLOW 001272	\$CDW2 020224	\$ERTTL 001110	\$OCNT 022766	\$TMP5 001206
UDPAR0= 177660	\$CHARC 022540	\$ESCAP 001214	\$OCTVL 024142	\$TMP6 001210
UDPAR1= 177662	\$CLR.T 022014	\$ETABL 020162	\$OMODE 022770	\$TN = 000032
UDPAR2= 177664	\$CMTAG 001100	\$ETEND 020266	\$OVER 020662	\$TPB 001150
UDPAR3= 177666	\$CM1 = 000006	\$FATAL 020144	\$PASS 020150	\$TPFLG 001155
UDPAR4= 177670	\$CM2 = 000014	\$FFLG 021574	\$PASTM 020006	\$TPS 001146
UDPAR5= 177672	\$CM3 = 000006	\$FILLC 001154	\$PWAD 023750	\$TRAP 023520
UDPAR6= 177674	\$CM4 = 000007	\$FILLS 001153	\$PWDRN 023610	\$TRAP2 023542
UDPAR7= 177676	\$CNTLG 023471	\$GDADR 001116	\$PWRMG 023744	\$TRP = 000016
UDPDR0= 177620	\$CNTLU 023464	\$GDDAT 001122	\$PWUP 023662	\$TRPAD 023554
UDPDR1= 177622	\$CPUOP 020170	\$GET42 021776	\$QUES 001222	\$TSTM 020004
UDPDR2= 177624	\$CRLF 001223	\$HD = 000000	\$RDCHR 023216	\$TSTM 001100
UDPDR3= 177626	\$DBLK 023206	\$HIBTS 020000	\$RDLIN 023346	\$TTYIN 023454
UDPDR4= 177630	\$DB20 024040	\$HIOCT 005736	\$RDOCT 005630	\$TYPDS 022772
UDPDR5= 177632	\$DDW0 020226	\$ICNT 001102	\$RDSZ = 000010	\$TYPE 022200
UDPDR6= 177634	\$DDW1 020230	\$ILLUP 023766	\$REGAD 001156	\$TYPEC 022412
UDPDR7= 177636	\$DDW10 020252	\$INTAG 001133	\$REG0 001160	\$TYPEX 022542
UIPAR0= 177640	\$DDW11 020254	\$ITEMB 001112	\$REG1 001162	\$TYPOC 022570
UIPAR1= 177642	\$DDW12 020256	\$LF 001224	\$REG2 001164	\$TYPON 022604
UIPAR2= 177644	\$DDW13 020260	\$LFLG 021573	\$REG3 001166	\$TYPOS 022544
UIPAR3= 177646	\$DDW14 020262	\$LOOP 022072	\$REG4 001170	\$UNIT 020154
UIPAR4= 177650	\$DDW15 020264	\$LPADR 001104	\$REG5 001172	\$UNITM 020010
UIPAR5= 177652	\$DDW2 020232	\$LPERR 001106	\$RESRE 022142	\$USWR 020166
UIPAR6= 177654	\$DDW3 020234	\$MADR1 020174	\$RTNAD 022074	\$VECT1 020212
UIPAR7= 177656	\$DDW4 020236	\$MADR2 020200	\$RTRN 022070	\$VECT2 020214
UIPDR0= 177600	\$DDW5 020240	\$MADR3 020204	\$SAVRE 022104	\$XOFF = 000023
UIPDR1= 177602	\$DDW6 020242	\$MADR4 020210	\$SAVR6 023772	\$XON = 000021
UIPDR2= 177604	\$DDW7 020244	\$MAIL 020142	\$SCOPE 020266	\$XTSTR 020372
UIPDR3= 177606	\$DDW8 020246	\$MAMS1 020172	\$SETUP= 000037	\$SET4= 000001
UIPDR4= 177610	\$DDW9 020250	\$MAMS2 020176	\$SSWR 000176	\$OFILL 022767
UIPDR5= 177612	\$DEVCT 020152	\$MAMS3 020202	\$STUP = 177777	\$.SX = 020000
UIPDR6= 177614	\$DEVM 020220	\$MAMS4 020206	\$SVLAD 020626	\$.SY = 002026
UIPDR7= 177616				

. ABS. 033727 000
000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 56440 WORDS (221 PAGES)
DYNAMIC MEMORY: 20034 WORDS (77 PAGES)
ELAPSED TIME: 00:10:24
CKKUAE.BIN,CKKUAE/CR/-SP/NL:TOC=CKKUAE.MLB/ML,CKKUAE.P11

SYMBOL	CROSS REFERENCE	REFERENCES								
SYMBOL	VALUE									
ABASE	= 000000	91-4095	91-4095							
ACDW1	= 000000	91-4095	91-4095							
ACDW2	= 000000	91-4095	91-4095							
ACPUOP	= 000000	91-4095	91-4095							
ADDROR	= 001236	#21-1496	*41-2228	*41-2243	*92-4097	*92-4097	106-4208	106-4209	106-4210	106-4213
ADDW0	= 177777	#89-4065	91-4095	91-4095						
ADDW1	= 177777	#89-4066	91-4095	91-4095						
ADDW10	= 000000	91-4095	91-4095							
ADDW11	= 000000	91-4095	91-4095							
ADDW12	= 000000	91-4095	91-4095							
ADDW13	= 000000	91-4095	91-4095							
ADDW14	= 000000	91-4095	91-4095							
ADDW15	= 000000	91-4095	91-4095							
ADDW2	= 000000	91-4095	91-4095							
ADDW3	= 000000	91-4095	91-4095							
ADDW4	= 000000	91-4095	91-4095							
ADDW5	= 000000	91-4095	91-4095							
ADDW6	= 000000	91-4095	91-4095							
ADDW7	= 000000	91-4095	91-4095							
ADDW8	= 000000	91-4095	91-4095							
ADDW9	= 000000	91-4095	91-4095							
ADEVCT	= 000000	91-4095	91-4095							
ADEVN	= 000000	91-4095	91-4095							
ADRAND	= 001232	#21-1496	*41-2230	*41-2245	*92-4097	*92-4097	106-4208	106-4209	106-4210	106-4213
ADREXT	= 003674	35-2057	#41-2220	56-2828	57-2897	59-2975	68-3332	68-3374	69-3445	
AENV	= 000000	91-4095	91-4095							
AENVN	= 000000	91-4095	91-4095							
AFATAL	= 000000	91-4095	91-4095							
AMADR1	= 000000	91-4095	91-4095							
AMADR2	= 000000	91-4095	91-4095							
AMADR3	= 000000	91-4095	91-4095							
AMADR4	= 000000	91-4095	91-4095							
AMAMS1	= 000000	91-4095	91-4095							
AMAMS2	= 000000	91-4095	91-4095							
AMAMS3	= 000000	91-4095	91-4095							
AMAMS4	= 000000	91-4095	91-4095							
AMSGAD	= 000000	91-4095	91-4095							
AMSLG	= 000000	91-4095	91-4095							
AMSGTY	= 000000	91-4095	91-4095							
AMTYP1	= 000000	91-4095	91-4095							
AMTYP2	= 000000	91-4095	91-4095							
AMTYP3	= 000000	91-4095	91-4095							
AMTYP4	= 000000	91-4095	91-4095							
APASS	= 000000	91-4095	91-4095							
APRIOR	= 000000	91-4095	91-4095							
APTCSU	= 000040	#94-4101	97-4107							
APTENV	= 000001	93-4099	94-4101	#94-4101	97-4107					
APTSIZ	= 000200	#94-4101								
APTSP0	= 000100	94-4101	#94-4101	97-4107						
ASWREG	= 000000	91-4095	91-4095							
ATESTN	= 000000	91-4095	91-4095							
AUNIT	= 000000	91-4095	91-4095							

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
AUSWR	=	000000	91-4095 91-4095
AVECT1	=	000000	91-4095 91-4095
AVECT2	=	000000	91-4095 91-4095
BADCPU		006607	#53-2625 54-2713
BADPC		001340	#21-1496 *36-2092 *37-2128 106-4205 106-4206 106-4207
BIT0	=	000001	#17-1489 38-2155 38-2160 58-2936 62-3090 63-3128 79-3745 79-3751
BIT00	=	000001	#17-1489 17-1489 50-2487 92-4097 92-4097 93-4099 93-4099
BIT01	=	000002	#17-1489 17-1489
BIT02	=	000004	#17-1489 17-1489
BIT03	=	000010	#17-1489 17-1489
BIT04	=	000020	#17-1489 17-1489
BIT05	=	000040	#17-1489 17-1489
BIT06	=	000100	#17-1489 17-1489 50-2505 50-2509
BIT07	=	000200	#17-1489 17-1489 50-2524
BIT08	=	000400	#17-1489 17-1489 50-2488 50-2540 92-4097
BIT09	=	001000	#17-1489 17-1489 55-2781 57-2909 59-2988 92-4097 93-4099
BIT1	=	000002	#17-1489
BIT10	=	002000	#17-1489 93-4099
BIT11	=	004000	#17-1489 42-2284 44-2364 46-2429 59-2957 68-3320 68-3339 73-3526 90-4078
BIT12	=	010000	#17-1489 92-4097 50-2489 50-2541 95-4103
BIT13	=	020000	#17-1489 93-4099
BIT14	=	040000	#17-1489 83-3851 84-3877 84-3882 85-3904 92-4097
BIT15	=	100000	#17-1489 45-2404
BIT2	=	000004	#17-1489
BIT3	=	000010	#17-1489
BIT4	=	000020	16-1460 #17-1489 32-1977 58-2937
BIT5	=	000040	#17-1489 34-2011 63-3127 69-3412 70-3473 72-3510 87-3974
BIT6	=	000100	#17-1489 78-3690
BIT7	=	000200	#17-1489
BIT8	=	000400	#17-1489
BIT9	=	001000	#17-1489 56-2834 61-3060 68-3349 69-3456 81-3823 83-3864 84-3891 85-3917
BPTVEC	=	000014	#17-1489
BUPWIN		001276	#21-1496 *65-3218 68-3367 68-3395
CACHE	=	177746	#16-1464 *50-2488 50-2489 *50-2502 *50-2505 *50-2509 50-2513 *50-2524 *50-2535
			*50-2540 50-2541 *77-3652 *78-3709
CACHVE	=	000114	#17-1489
CASHSR		005324	#50-2487 77-3651 78-3701
CASH1		005460	#50-2512 *77-3649 *78-3699
CASH2		005470	#50-2516 *77-3650 *78-3700
CHARCT		005760	#52-2599
CHKLMA		005076	38-2168 42-2281 44-2361 #46-2425 68-3326 68-3345 68-3354
CHKPAT		005600	#50-2547 56-2817 56-2831
CLRMAP		002772	#34-2008 57-2882 60-3012 68-3284
CMPE	=	177744	#16-1465 *50-2510 *50-2534
CNTR		001322	#21-1496
CONTRL	=	177746	#17-1489
CPFLAG		003224	#36-2083 *36-2099 *36-2102 *36-2107 *54-2740 *93-4099
CPSAVE		020700	29-1900 *92-4097 92-4097 #92-4097 *93-4099 93-4099
CPUER		003222	#36-2082 54-2734 55-2787
CPUERR	=	177766	#17-1489 35-2042 *35-2043 36-2091 *36-2108 *54-2743 *93-4099
CPUEXP		001326	#21-1496 35-2039 35-2044 35-2053 36-2093 36-2097 36-2106 *44-2345 *44-2350

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
CPUMSG	006200	*44-2353	*44-2355	*54-2736	*59-2962	*59-2964	*59-2967	*59-2969	*59-2977	*59-2979
CPUTYP	007140	*59-2990	*59-2992	*60-3017	*60-3019	*60-3032	*60-3034	*60-3044	*61-3064	*62-3098
CR	= 000015	*62-3101	*63-3137	*63-3140	*63-3148	*64-3173	*64-3179	*64-3182	*65-3209	*65-3212
CRLF	= 000200	*65-3215	*68-3312	*68-3314	*90-4082	*90-4087	106-4205	106-4214	106-4229	
		#53-2620	*54-2692	54-2693	*54-2698	54-2699				
		35-2036	35-2049	36-2095	#53-2630	*54-2684	*54-2685	54-2695	55-2755	77-3638
		#17-1489	97-4107	97-4107						
		#17-1489	52-2617	52-2619	53-2620	53-2620	53-2621	53-2622	53-2623	53-2624
		53-2625	53-2626	53-2628	53-2629	97-4107	97-4107	104-4134	104-4138	104-4139
		104-4162	104-4164	105-4171	105-4173	105-4175	105-4177	105-4180	105-4182	105-4189
		105-4193	105-4198	105-4200	107-4232	107-4232	107-4233	107-4234	107-4235	107-4236
		107-4237	107-4237	107-4238	107-4239	107-4240				
CTRAPS	= 000116	#16-1468	50-2496	*50-2498	*50-2539					
CTRAPV	= 000114	#16-1467	50-2495	*50-2497	*50-2538					
DATA	001364	#21-1496	44-2346	*44-2381	*44-2388	*44-2389	*44-2390			
DATAAND	001242	#21-1496	*39-2187	*56-2861	*92-4097	106-4209	106-4210	106-4212	106-4213	
DATAOR	001246	#21-1496	*39-2185	*56-2859	*92-4097	106-4209	106-4210	106-4212	106-4213	
DATEXT	003624	#39-2185	56-2824	57-2901	61-3051	68-3335	69-3448			
DATO	= 100000	#84-3874	84-3881							
DATOB	= 140000	#85-3901	85-3908							
DDISP	= 177570	#17-1489	21-1496							
DFMSG	006120	52-2597	#52-2616							
DF1	033641	22-1501	22-1507	22-1514	23-1548	23-1561	23-1569	23-1583	24-1595	24-1601
		24-1608	24-1614	24-1626	24-1632	24-1638	24-1644	25-1649	27-1723	27-1729
		#108-4242								
DF11	033671	23-1555	#108-4246							
DF14	033677	23-1576	26-1684	#108-4247						
DF201	033711	26-1658	26-1671	26-1690	26-1696	27-1709	27-1717	#108-4249		
DF204	033717	26-1677	#108-4250							
DF210	033724	26-1702	#108-4251							
DF23	033705	24-1620	#108-4248							
DF4	033646	22-1521	#108-4243							
DF5	033653	22-1528	23-1590	#108-4244						
DF6	033661	22-1535	23-1541	26-1665	#108-4245					
DH1	030121	22-1499	#105-4169							
DH10	030727	23-1546	23-1559	23-1567	24-1593	24-1599	24-1606	24-1612	#105-4179	
DH11	030754	23-1552	23-1573	23-1587	#105-4180					
DH15	031111	23-1580	#105-4182							
DH2	030160	22-1505	#105-4170							
DH201	031366	26-1656	#105-4188							
DH202	031415	26-1662	#105-4189							
DH203	031524	26-1669	#105-4191							
DH204	031605	26-1675	#105-4192							
DH205	031656	26-1681	#105-4193							
DH206	031743	26-1688	#105-4195							
DH207	032013	26-1694	#105-4196							
DH210	032054	26-1700	#105-4197							
DH211	032106	27-1706	27-1714	#105-4198						
DH212	032211	#105-4200								
DH213	032270	27-1721	27-1727	#105-4202						
DH23	031170	24-1618	#105-4184							
DH24	031231	24-1624	24-1630	24-1636	#105-4185					

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
DH27		031270	24-1642 #105-4186
DH3		030207	22-1511 #105-4171
DH30		031327	25-1647 #105-4187
DH4		030306	22-1518 #105-4173
DH5		030403	22-1525 #105-4175
DH6		030540	22-1532 23-1538 #105-4177
DISPLA		001140	#21-1496 47-2447 92-4097 93-4099
DISPRE		000174	#18-1492
DSABLD		005230	#48-2459 63-3141 64-3183 65-3207
DSWR	=	177570	#17-1489 21-1496
DTMS		005740	#52-2597 66-3227
DTMSG		005744	52-2597 #52-2598
DT1		032340	22-1500 #106-4205
DT10		032452	23-1547 23-1560 23-1568 24-1594 24-1600 24-1607 24-1613 #106-4211
DT11		032462	23-1554 #106-4212
DT14		032500	23-1575 23-1589 #106-4213
DT15		032516	23-1582 #106-4214
DT2		032352	22-1506 #106-4206
DT201		032612	26-1657 #106-4220
DT202		032622	26-1664 #106-4221
DT203		032636	26-1670 #106-4222
DT204		032654	26-1676 #106-4223
DT205		032670	26-1683 #106-4224
DT206		032702	26-1689 #106-4225
DT207		032716	26-1695 #106-4226
DT210		032730	26-1701 #106-4227
DT211		032740	27-1708 27-1716 #106-4228
DT212		032754	#106-4229
DT213		032766	27-1722 27-1728 #106-4230
DT23		032530	24-1619 #106-4215
DT24		032542	24-1625 24-1631 #106-4216
DT26		032554	24-1637 #106-4217
DT27		032566	24-1643 #106-4218
DT3		032362	22-1513 #106-4207
DT30		032600	25-1648 #106-4219
DT4		032376	22-1520 #106-4208
DT5		032412	22-1527 #106-4209
DT6		032430	22-1534 23-1540 #106-4210
EADRES		005764	*41-2235 *41-2236 41-2247 *41-2249 *41-2250 *42-2292 *42-2293 *44-2371 *44-2372 #52-2601 *59-2987 *79-3762 *79-3763 *88-4045 *92-4097 106-4215 106-4220 106-4221 106-4224 106-4225 106-4226 106-4227 106-4228
EADRS2		005770	#52-2602 *57-2902 *69-3449 *69-3450 *79-3760 *79-3761 *92-4097 106-4215 106-4221 106-4222 106-4224 54-2655
EMTVEC	=	000030	#17-1489
EM1		024160	22-1498 #104-4127
EM10		025003	23-1544 #104-4134
EM11		025155	23-1551 #104-4136
EM12		025247	23-1558 #104-4137
EM13		025321	23-1564 #104-4138
EM14		025525	23-1572 #104-4141
EM15		025630	23-1579 #104-4142
EM16		025723	23-1586 #104-4143

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
EM17		026021	24-1592 #104-4144
EM2		024245	22-1504 #104-4128
EM20		026051	24-1598 #104-4145
EM201		026625	26-1655 #104-4154
EM202		026701	26-1661 #104-4155
EM203		026774	26-1668 #104-4156
EM204		027055	26-1674 #104-4157
EM205		027124	26-1680 #104-4158
EM206		027206	26-1687 #104-4159
EM207		027237	26-1693 #104-4160
EM21		026150	24-1604 #104-4146
EM210		027346	26-1699 #104-4161
EM211		027446	27-1704 #104-4162
EM212		027604	27-1712 #104-4164
EM213		027737	27-1720 #104-4166
EM214		030030	27-1726 #104-4167
EM22		026226	24-1611 #104-4147
EM23		026314	24-1617 #104-4148
EM24		026344	24-1623 #104-4149
EM25		026402	24-1629 #104-4150
EM26		026437	24-1635 #104-4151
EM27		026472	24-1641 #104-4152
EM3		024317	22-1510 #104-4129
EM30		026572	25-1646 #104-4153
EM4		024425	22-1517 #104-4130
EM5		024505	22-1524 #104-4131
EM6		024600	22-1531 #104-4132
EM7		024702	23-1537 #104-4133
ENABLD		005266	#49-2474 63-3149 64-3177 65-3216
ERRCNT		001320	#21-1496 29-1791 29-1798 *35-2051 *36-2105 *55-2757 55-2759 *55-2778 55-2783
			55-2789 *55-2792 *56-2809 56-2841 *56-2845 *56-2862 57-2915 *57-2918 61-3065
			*61-3068 *68-3291 *68-3315 68-3398 *68-3401 69-3459 *69-3462 *87-3986 *90-4074
			*92-4097 *93-4099 106-4208 106-4209 106-4210 106-4212 106-4213
ERROR	=	104000	#16-1473 #17-1489 35-2047 35-2059 36-2100 36-2103 37-2134 55-2793 56-2833
			56-2848 56-2850 57-2908 57-2919 59-2986 60-3025 61-3059 61-3069 61-3072
			62-3105 63-3142 63-3146 63-3150 64-3178 64-3185 65-3208 65-3217 68-3348
			68-3375 68-3402 69-3455 69-3463 70-3478 71-3493 72-3515 74-3570 75-3588
			76-3601 77-3655 78-3704 78-3708 79-3765 80-3786 81-3802 81-3822 82-3839
			83-3863 84-3890 85-3916 87-3984 88-4046
ERRVEC	=	000004	#17-1489 *54-2734 *54-2735 *55-2754 *55-2787 92-4097 *92-4097 *92-4097 *92-4097
ERTYPE		002026	#29-1777 93-4099
ER200		001666	#26-1650 29-1806 29-1809
EXTOUT		005762	#52-2600
FJBIT	=	000100	#80-3779 80-3782 80-3785 81-3797 81-3798 81-3801 82-3834 82-3835 82-3838
FLAG		001324	#21-1496
FLOATR		005774	38-2150 *38-2155 *38-2160 *38-2162 48-2463 49-2478 #52-2603
FTTHRU		003532	#38-2158 *63-3133
GMRMD1		007043	#53-2628 *55-2768 55-2769 55-2770
GMRMOD		007046	#53-2629 55-2772
GNS	=	*****	18-1492 18-1492 95-4103 95-4103 101-4115 101-4115 101-4115 101-4115 101-4115
			101-4115 101-4115 101-4115 101-4115 101-4115 101-4115 101-4115 101-4115 101-4115
			101-4115 101-4115 101-4115 101-4115 101-4115 101-4116 101-4116 101-4117

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	101-4117	101-4118	101-4118						
HIADRS	= 177742	#17-1489									
HIGEST	= 001260	#21-1496	*64-3167	*64-3188	64-3189	65-3199	68-3362	68-3388	81-3805	87-3979	
HITMIS	= 177752	#17-1489									
HT	= 000011	#17-1489	97-4107	97-4107							
IBSAVE	= 020702	#93-4099	*93-4099	93-4099	*93-4099	*93-4099	93-4099	93-4099	93-4099	93-4099	
IOTVEC	= 000020	#17-1489	54-2655	54-2655							
JMPMSG	= 033002	66-3226	#107-4232								
KDPAR0	= 172360	#17-1489									
KDPAR1	= 172362	#17-1489									
KDPAR2	= 172364	#17-1489									
KDPAR3	= 172366	#17-1489									
KDPAR4	= 172370	#17-1489									
KDPAR5	= 172372	#17-1489									
KDPAR6	= 172374	#17-1489									
KDPAR7	= 172376	#17-1489									
KDPDR0	= 172320	#17-1489									
KDPDR1	= 172322	#17-1489									
KDPDR2	= 172324	#17-1489									
KDPDR3	= 172326	#17-1489									
KDPDR4	= 172330	#17-1489									
KDPDR5	= 172332	#17-1489									
KDPDR6	= 172334	#17-1489									
KDPDR7	= 172336	#17-1489									
KERSTK	= 001100	#17-1489									
KIPAR0	= 172340	#17-1489	30-1922	58-2929	*73-3555	*77-3636	*79-3750	*87-3990			
KIPAR1	= 172342	#17-1489	*73-3553	*77-3637	*87-3991						
KIPAR2	= 172344	#17-1489	69-3447	79-3747	*79-3748	*79-3764	*79-3766	*79-3771	83-3847	*83-3848	
		#17-1489	*83-3866	84-3875	*84-3876	*84-3893	85-3902	*85-3903	*85-3919		
KIPAR3	= 172346	#17-1489	45-2401	*45-2402	*45-2407	54-2719	*54-2720	*54-2725	54-2726	*54-2728	
KIPAR4	= 172350	#17-1489	*44-2338	44-2378	*44-2382	57-2896	*63-3130	*63-3143	63-3144	64-3158	
		64-3165	*64-3171	*64-3174	64-3175	64-3188	*65-3204	65-3205	65-3218	65-3219	
		*68-3292	*68-3305	*68-3306	68-3309	68-3324	68-3331	68-3343	68-3358	68-3360	
		68-3362	*68-3364	68-3365	68-3367	*69-3413	*69-3420	69-3421	81-3809	*81-3810	
		*81-3814	81-3815	*81-3826	*90-4075	*90-4080	106-4218				
KIPAR5	= 172352	#17-1489	*60-3027	*60-3029	61-3070	*61-3073	*62-3094	106-4219			
KIPAR6	= 172354	#17-1489	*42-2275	*44-2339	59-2953	*59-2954	*59-2959	59-2961	59-2974	*59-2993	
		*59-2995	*59-2998	*60-3014	*60-3022	60-3027	*60-3039	62-3091	69-3426	69-3429	
		*69-3439	69-3440	69-3444	*88-4018	*88-4025	*88-4027	*88-4052			
KIPAR7	= 172356	#17-1489	35-2056	56-2827	57-2899	68-3334	68-3373				
KIPDR0	= 172300	#17-1489	58-2924								
KIPDR1	= 172302	#17-1489									
KIPDR2	= 172304	#17-1489									
KIPDR3	= 172306	#17-1489									
KIPDR4	= 172310	#17-1489									
KIPDR5	= 172312	#17-1489									
KIPDR6	= 172314	#17-1489									
KIPDR7	= 172316	#17-1489									
KSP	=%000006	#17-1489	*29-1777	*29-1781	29-1794	29-1802	*29-1831	*29-1839	*29-1845	*29-1851	
		29-1853	*29-1854	*29-1862	*29-1878	*29-1885	*29-1891	*29-1892	30-1915	*30-1930	
		30-1932	*30-1933	*30-1938	30-1938	31-1948	*31-1953	31-1955	*31-1956	*31-1960	
		31-1960	32-1978	32-1980	32-1981	33-1993	*35-2034	*35-2035	*35-2061	*35-2062	

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
MAPH32	=	170352	#17-1489
MAPH33	=	170356	#17-1489
MAPH34	=	170362	#17-1489
MAPH35	=	170366	#17-1489
MAPH36	=	170372	#17-1489
MAPH37	=	170376	#17-1489 34-2015 56-2846 56-2852 57-2880 57-2888
MAPH4	=	170222	#17-1489
MAPH5	=	170226	#17-1489
MAPH6	=	170232	#17-1489
MAPH7	=	170236	#17-1489
MAPLO	=	170200	#17-1489 34-2008 55-2758 55-2776 *59-2956 59-2970 *59-2972 59-2983 *59-2985 63-3119
MAPL00	=	170200	#17-1489 17-1489 56-2811 57-2878
MAPL01	=	170204	#17-1489 17-1489 68-3293 68-3377
MAPL02	=	170210	#17-1489 17-1489
MAPL03	=	170214	#17-1489 17-1489
MAPL04	=	170220	#17-1489 17-1489
MAPL05	=	170224	#17-1489 17-1489
MAPL06	=	170230	#17-1489 17-1489
MAPL07	=	170234	#17-1489 17-1489
MAPL1	=	170204	#17-1489
MAPL10	=	170240	#17-1489
MAPL11	=	170244	#17-1489
MAPL12	=	170250	#17-1489
MAPL13	=	170254	#17-1489
MAPL14	=	170260	#17-1489
MAPL15	=	170264	#17-1489
MAPL16	=	170270	#17-1489
MAPL17	=	170274	#17-1489
MAPL2	=	170210	#17-1489
MAPL20	=	170300	#17-1489
MAPL21	=	170304	#17-1489
MAPL22	=	170310	#17-1489
MAPL23	=	170314	#17-1489
MAPL24	=	170320	#17-1489
MAPL25	=	170324	#17-1489
MAPL26	=	170330	#17-1489
MAPL27	=	170334	#17-1489
MAPL3	=	170214	#17-1489
MAPL30	=	170340	#17-1489
MAPL31	=	170344	#17-1489
MAPL32	=	170350	#17-1489
MAPL33	=	170354	#17-1489
MAPL34	=	170360	#17-1489
MAPL35	=	170364	#17-1489
MAPL36	=	170370	#17-1489
MAPL37	=	170374	#17-1489
MAPL4	=	170220	#17-1489
MAPL5	=	170224	#17-1489
MAPL6	=	170230	#17-1489
MAPL7	=	170234	#17-1489
MASK1		005776	50-2548 #52-2604 *56-2812 56-2825 *56-2856

CKKUAE CREATED BY MACRO ON 11-JAN-82 AT 11:32 PAGE 9
SEQUENCE 147 CREF V01

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
MASK2	=	006000	#17-1489	50-2551	#52-2605	*56-2813	*56-2857					
MEMERR	=	177744	#16-1472	54-2683								
MFPT	=	000007	#37-2122	*37-2136		*54-2742	*93-4099					
MMFLAG		003374	#21-1496	52-2598		*64-3168	*64-3186	*64-3187				
MMRHI		001270	#21-1496	52-2598		*64-3157						
MMRLOW		001256	#17-1489	17-1489		37-2131	*37-2135	*50-2487	*54-2737	*57-2877	*58-2936	*62-3090
MMRO	=	177572	*63-3128	*68-3285		*68-3290	*77-3634	*79-3745	*79-3751	*83-3849	*87-3973	*88-4020
			*93-4099	*95-4103								
MMR1	=	177574	#17-1489	17-1489		37-2132						
MMR2	=	177576	#17-1489	17-1489		37-2133						
MMR3	=	172516	#17-1489	17-1489		34-2011	*54-2738	*57-2884	*58-2937	*63-3127	*68-3281	*68-3283
			*69-3412	*70-3473		*77-3635	*79-3746	*79-3749	*80-3780	*81-3807	*81-3827	*83-3850
			*87-3988	*87-3989		*95-4103	106-4211					
			70-3477	74-3569		#90-4071						
MMTOTM		020014	#37-2121	54-2732								
MMTRAP		003372	#17-1489	*54-2732		*54-2733						
MMVEC	=	000250	#53-2627	55-2766								
MRQUES		006754	#52-2618	54-2705								
NEWSWR		006161	44-2353	44-2356		#72-3510						
NEXMEM	=	000040	38-2153	38-2156		38-2161	#38-2163					
NEXT		003552	29-1796	#53-2623								
NOMORE		006421	64-3190	66-3225		66-3235	#67-3253					
NOMSG		013646	#21-1496	*87-3995		88-4019						
NUMOFK		001316	#21-1496	*47-2444		*87-3976						
NXTTST		001362	#21-1496	*35-2034		35-2062	*36-2089	36-2092	36-2110	*37-2129	37-2138	
OLDPC		001342	#21-1496	*35-2035		35-2061	*36-2090	36-2109	*37-2130	37-2137		
OLDPS		001344	#21-1496	*32-1980		33-1993	*33-1994					
OLDPSW		001346	#21-1496	*30-1928		*31-1952						
PADRSR		001230	#21-1496	*30-1929		30-1930	*31-1951	31-1953				
PADRSL		001226	#21-1496	*40-2201		*56-2860	*92-4097	106-4210	106-4212			
PATAND		001252	#53-2634									
PATCH		007142	#40-2199	56-2822		61-3053						
PATEXT		003650	#52-2608	56-2816		60-3041						
PATRNS		006102	#21-1496	*40-2199		*56-2858	*92-4097	106-4210	106-4212			
PATTOR		001254	#21-1496									
PCONTR		001334	#21-1496	*35-2038		*35-2042	35-2043	35-2044	*36-2091	36-2097	*36-2106	36-2108
PCPUER		001330	*38-2164	38-2171		*44-2341	44-2351	44-2356	*60-3016	60-3020	*68-3307	68-3316
			*87-3987	88-4043		*88-4049	*90-4081	90-4088	106-4205	106-4206	106-4214	106-4229
PIRQ	=	177772	#17-1489									
PIRQVE	=	000240	#17-1489									
PMAINT		001336	#21-1496									
PMBECD		002526	29-1896	#29-1900								
PMBECF		002536	29-1896	#29-1901								
PMBECH		002476	29-1896	#29-1898								
PMBECM		002442	29-1896	#29-1897								
PMBECW		002432	29-1786	#29-1896								
PMMRO		001350	#21-1496	*37-2131		106-4207						
PMMR1		001352	#21-1496	*37-2132		106-4207						
PMMR2		001354	#21-1496	*37-2133		106-4207						
PPARER		001332	#21-1496									
PRETST		005166	#47-2443	55-2753		56-2808	57-2876	59-2952	60-3011	62-3088	68-3280	69-3411

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
SDPDR7	=	172236	#17-1489
SIPAR0	=	172240	#17-1489
SIPAR1	=	172242	#17-1489
SIPAR2	=	172244	#17-1489
SIPAR3	=	172246	#17-1489
SIPAR4	=	172250	#17-1489
SIPAR5	=	172252	#17-1489
SIPAR6	=	172254	#17-1489
SIPAR7	=	172256	#17-1489
SIPDR0	=	172200	#17-1489
SIPDR1	=	172202	#17-1489
SIPDR2	=	172204	#17-1489
SIPDR3	=	172206	#17-1489
SIPDR4	=	172210	#17-1489
SIPDR5	=	172212	#17-1489
SIPDR6	=	172214	#17-1489
SIPDR7	=	172216	#17-1489
SIZEH1	=	177762	#17-1489
SIZEJ0	=	012744	60-3026 62-3089 62-3104 #63-3118 87-3981
SIZEJ1	=	013154	#64-3157
SIZEJ2	=	013402	#65-3199
SIZELO	=	177760	#17-1489
SPECST	=	006002	#52-2606 63-3121 63-3126 66-3234 66-3237
SR0	=	177572	#17-1489
SR1	=	177574	#17-1489
SR2	=	177576	#17-1489
SR3	=	172516	#17-1489
SSP	=	%000006	#17-1489
STACK	=	001100	#17-1489 17-1489 17-1489 17-1489 54-2648 54-2666
START	=	010000	18-1492 #54-2642 63-3147 102-4120
STKLMT	=	177774	#17-1489
SUPSTK	=	000700	#17-1489
SWR	=	001136	#21-1496 29-1800 42-2284 44-2364 46-2429 *54-2689 *54-2691 *54-2697 *54-2702 *54-2731 55-2781 56-2834 57-2909 59-2988 61-3060 68-3320 68-3339 68-3349 69-3456 73-3526 77-3647 78-3690 78-3697 81-3823 83-3864 84-3891 85-3917 87-3974 90-4078 92-4097 92-4097 92-4097 92-4097 92-4097 93-4099 93-4099 93-4099 93-4099 93-4099 95-4103 102-4120 102-4120
SWREG	=	000176	#18-1492
SW0	=	000001	#17-1489
SW00	=	000001	#17-1489 17-1489
SW01	=	000002	#17-1489 17-1489
SW02	=	000004	#17-1489 17-1489
SW03	=	000010	#17-1489 17-1489
SW04	=	000020	#17-1489 17-1489
SW05	=	000040	#17-1489 17-1489
SW06	=	000100	#17-1489 17-1489
SW07	=	000200	#17-1489 17-1489
SW08	=	000400	#17-1489 17-1489
SW09	=	001000	#17-1489 17-1489
SW1	=	000002	#17-1489
SW10	=	002000	#17-1489
SW11	=	004000	#17-1489

SYMBOL	CROSS REFERENCE	REFERENCES
SYMBOL	VALUE	
SW12	= 010000	#17-1489
SW13	= 020000	#17-1489
SW14	= 040000	#17-1489
SW15	= 100000	#17-1489
SW2	= 000004	#17-1489
SW3	= 000010	#17-1489
SW4	= 000020	#17-1489
SW5	= 000040	#17-1489
SW6	= 000100	#17-1489
SW7	= 000200	#17-1489 29-1800
SW8	= 000400	#17-1489
SW9	= 001000	#17-1489 93-4099
SYSTID	= 177764	#17-1489
TBIT	= 000020	#32-1977 32-1978 32-1981 33-1994
TBITO	= 104413	#101-4116
TBITOF	002726	#32-1978 101-4116
TBITR	= 104414	58-2920 73-3525 95-4103 #101-4117
TBITRE	002754	#33-1993 101-4117
TBITVE	= 000014	#17-1489 54-2656
TCPMRA	004416	#44-2329 72-3514 76-3600
TIMEOU	003030	#35-2027 55-2754
TIMOUT	= 000020	#16-1460 35-2053 59-2962 59-2967 59-2977 59-2990 60-3017 60-3032 60-3044 62-3098 63-3137 64-3179 65-3209 68-3312 90-4082 90-4088
TKVEC	= 000060	#17-1489
TOFLAG	003032	#35-2028 *35-2046 *35-2058 *35-2060 *54-2741 *93-4099
TOMSG	006261	#53-2621 66-3236
TPVEC	= 000064	#17-1489
TRAPVE	= 000034	#17-1489 54-2655 54-2656
TRTVEC	= 000014	#17-1489
TSTLOC	003474	#38-2149 63-3138 64-3180 65-3210 81-3817 81-3820
TST1	010676	54-2669 54-2670 #55-2753
TST10	014616	68-3280 68-3399 #69-3411
TST11	015072	69-3411 69-3428 69-3460 #70-3472
TST12	015140	70-3472 #71-3489
TST13	015174	71-3489 #72-3503
TST14	015334	#74-3567
TST15	015362	74-3567 #75-3584
TST16	015420	75-3584 #76-3598
TST17	015450	76-3598 #77-3633
TST2	011122	55-2753 55-2790 #56-2808
TST20	015610	77-3633 77-3645 77-3648 77-3654 #78-3689
TST21	015744	73-3529 77-3641 78-3689 #79-3724
TST22	016220	79-3724 #80-3778
TST23	016266	80-3778 80-3783 #81-3796
TST24	016474	81-3796 81-3806 #82-3833
TST25	016540	82-3833 82-3836 #83-3846
TST26	016702	83-3846 #84-3873
TST27	017036	84-3873 #85-3900
TST3	011444	56-2808 56-2851 56-2853 #57-2876
TST30	017164	78-3711 85-3900 #87-3972
TST31	017412	87-3972 87-4001 #88-4015
TST32	= 021576	#87-4008 88-4015

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES
UIPDR1	=	177602	#17-1489
UIPDR2	=	177604	#17-1489
UIPDR3	=	177606	#17-1489
UIPDR4	=	177610	#17-1489
UIPDR5	=	177612	#17-1489
UIPDR6	=	177614	#17-1489
UIPDR7	=	177616	#17-1489
USESTK	=	000600	#17-1489
USP	=	%000006	#17-1489
YESMSG		013542	64-3169 #66-3224
\$APTHD		020000	89-4067 #89-4067
\$ASTAT	=	*****	94-4101 94-4101
\$ATYC		021354	94-4101 #94-4101
\$ATY1		021330	#94-4101
\$ATY3		021336	#94-4101 97-4107
\$ATY4		021346	93-4099 #94-4101
\$AUTOB		001132	#21-1496
\$BASE		020216	#91-4095
\$BDADR		001120	#21-1496
\$BDDAT		001124	#21-1496
\$BELL		001216	#21-1496 93-4099 93-4099 93-4099
\$CDW1		020222	#91-4095
\$CDW2		020224	#91-4095
\$CHARC		022540	*97-4107 *97-4107 97-4107 *97-4107 #97-4107
\$CKSWR	=	*****	101-4115
\$CLR.T		022014	95-4103 #95-4103
\$CMTAG		001100	#21-1496 54-2643 54-2644
\$CM1	=	000006	#21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496
\$CM2	=	000014	#21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496
\$CM3	=	000006	#21-1496 21-1496 21-1496
\$CM4	=	000007	#21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496 #21-1496 21-1496 21-1496
\$CNTLG		023471	#100-4113
\$CNTLU		023464	#100-4113
\$CPUOP		020170	#91-4095
\$CRLF		001223	#21-1496 29-1813 29-1819 29-1826 66-3232 66-3233 66-3240 66-3245 93-4099
\$DBLK		023206	93-4099 99-4111 99-4111 #99-4111
\$DB20		024040	29-1852 30-1931 31-1954 #103-4124
\$DDW0		020226	63-3118 #91-4095
\$DDW1		020230	38-2152 #91-4095
\$DDW10		020252	#91-4095
\$DDW11		020254	#91-4095
\$DDW12		020256	#91-4095
\$DDW13		020260	#91-4095
\$DDW14		020262	#91-4095
\$DDW15		020264	#91-4095

SYMBOL	CROSS REFERENCE VALUE	REFERENCES									
\$DDW2	020232	#91-4095									
\$DDW3	020234	#91-4095									
\$DDW4	020236	#91-4095									
\$DDW5	020240	#91-4095									
\$DDW6	020242	#91-4095									
\$DDW7	020244	#91-4095									
\$DDW8	020246	#91-4095									
\$DDW9	020250	#91-4095									
\$DEVCT	020152	#91-4095									
\$DEVN	020220	#91-4095									
\$DOAGN	022034	95-4103	95-4103	95-4103	#95-4103						
\$DTBL	023176	99-4111	#99-4111								
\$ENDAD	022024	19-1494	93-4099	#95-4103							
\$ENDCT	021644	54-2657	#95-4103								
\$ENULL	022100	#95-4103									
\$ENV	020162	48-2459	49-2474	55-2761	#91-4095	93-4099	94-4101	94-4101	97-4107		
\$ENVN	020163	48-2461	49-2476	54-2700	54-2729	77-3642	78-3692	#91-4095	94-4101	97-4107	
		97-4107									
\$EOP	021576	87-3976	87-3985	87-4008	88-4057	#95-4103					
\$EOPCT	021636	*54-2657	#95-4103	95-4103							
\$ERFLG	001101	#21-1496	92-4097	*92-4097	92-4097	92-4097	*92-4097	92-4097	92-4097	*93-4099	
		93-4099	93-4099								
\$ERMAX	001113	#21-1496	*54-2659	92-4097	*92-4097	92-4097	92-4097				
\$ERROR	020704	54-2655	#93-4099								
\$ERRPC	001114	#21-1496	29-1781	29-1900	*93-4099	*93-4099	93-4099	93-4099	93-4099	106-4208	
		106-4211	106-4214	106-4215	106-4216	106-4217	106-4218	106-4219	106-4220	106-4221	
		106-4222	106-4223	106-4224	106-4225	106-4226	106-4227	106-4228	106-4229	106-4230	
\$ERRTB	001366	#22-1496	29-1809	29-1814							
\$ERTTL	001110	#21-1496	*55-2791	*56-2844	*57-2917	*61-3067	*68-3400	*69-3461	*88-4048	*93-4099	
		93-4099	93-4099	95-4103	95-4103	*95-4103					
\$ESCAP	001214	#21-1496	*54-2658	*92-4097	93-4099	93-4099	93-4099	93-4099			
\$ETABL	020162	#91-4095									
\$ETEND	020266	89-4067	#91-4095								
\$FATAL	020144	#91-4095	*94-4101								
\$FFLG	021574	*94-4101	*94-4101	94-4101	*94-4101	#94-4101					
\$FILLC	001154	#21-1496	97-4107	97-4107	97-4107						
\$FILLS	001153	#21-1496	97-4107	97-4107							
\$GDADR	001116	#21-1496									
\$GDDAT	001122	#21-1496									
\$GET42	021776	#95-4103									
\$GTSWR	= *****	101-4115									
\$HD	= 000000	16-1485	16-1485	16-1485							
\$HIBTS	020000	#89-4067									
\$HIOCT	005736	*51-2590	#51-2595								
\$ICNT	001102	#21-1496	*92-4097	92-4097	*92-4097	92-4097	92-4097				
\$ILLUP	023766	102-4120	102-4120	#102-4120							
\$INTAG	001133	#21-1496									
\$ITEMB	001112	#21-1496	29-1779	*93-4099	93-4099	93-4099	*93-4099	93-4099	93-4099	93-4099	
\$LF	001224	#21-1496	93-4099	93-4099	97-4107	97-4107	100-4113	100-4113	100-4113		
\$LFLG	021573	*94-4101	#94-4101								
\$LOOP	022072	95-4103	#95-4103								
\$LPADR	001104	#21-1496	*47-2449	*54-2669	*92-4097	*92-4097	92-4097	92-4097	92-4097		

SYMBOL	CROSS REFERENCE	VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
\$RDLIN		023346	#100-4113	101-4115	101-4115							
\$RDOCT		005630	#51-2565	101-4115	101-4115		101-4118					
\$RDSZ	=	000010	#100-4113	100-4113								
\$REGAD		001156	#21-1496									
\$REGO		001160	#21-1496	*93-4099	106-4226							
\$REG1		001162	#21-1496	*93-4099	106-4216							
\$REG2		001164	#21-1496	*93-4099	106-4217	106-4223	106-4228					
\$REG3		001166	#21-1496	*93-4099	106-4222	106-4228						
\$REG4		001170	#21-1496	*93-4099	106-4222							
\$REG5		001172	#21-1496	*93-4099	106-4230							
\$RESRE		022142	#96-4105	101-4115								
\$RTNAD		022074	#95-4103									
\$RTRN		022070	54-2656	*54-2660	*54-2665	95-4103	#95-4103					
\$R2A	=	*****	101-4115									
\$SAVRE		022104	#96-4105	101-4115	101-4115							
\$SAVR6		023772	*102-4120	102-4120	*102-4120	*102-4120	#102-4120					
\$SCOPE		020266	54-2655	#92-4097								
\$SETUP	=	000037	#18-1491	18-1491	#18-1491	18-1491	#18-1491	18-1491	#18-1491	18-1491	#18-1491	#18-1491
			18-1491	#18-1491	92-4097	93-4099	93-4099	93-4099	93-4099	95-4103	95-4103	
			100-4113	100-4113	102-4120							
\$SSWR		000176	#28-1733	54-2697	54-2702	54-2703	*54-2707	*54-2709				
\$STUP	=	177777	#18-1491	#18-1491	18-1491	#18-1491	#18-1491	18-1491	#18-1491	#18-1491	18-1491	
			#18-1491	#18-1491	18-1491	#18-1491	#18-1491	18-1491	#18-1491	#18-1491	18-1491	
\$SVLAD		020626	92-4097	#92-4097								
\$SVPC	=	000204	#19-1494	19-1494								
\$SWR	=	177400	#16-1470	16-1485	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486
			16-1486	21-1496	21-1496	21-1496	55-2753	56-2808	57-2876	59-2952	60-3011	
			62-3087	68-3280	69-3411	70-3472	71-3489	72-3503	74-3567	75-3584	76-3598	
			77-3633	78-3689	79-3724	80-3778	81-3796	82-3833	83-3846	84-3873	85-3900	
			87-3972	88-4015	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097
			92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097
			92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097
			93-4099	93-4099	93-4099	93-4099	93-4099	93-4099	93-4099	93-4099	93-4099	93-4099
			95-4103	95-4103	102-4120							
\$SWREG		020164	54-2731	77-3644	78-3694	#91-4095						
\$SWRMK	=	000340	#16-1469	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486	16-1486
			16-1486	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097	92-4097
			92-4097	92-4097	92-4097							
\$TBIT		022076	*54-2668	*95-4103	95-4103	95-4103	#95-4103	*102-4120				
\$TESTN		020146	29-1900	52-2598	*87-3983	#91-4095	*92-4097	*93-4099	106-4205	106-4206	106-4207	
			106-4208	106-4209	106-4210	106-4211	106-4212	106-4213	106-4214	106-4215	106-4216	
			106-4217	106-4218	106-4219	106-4220	106-4221	106-4222	106-4223	106-4224	106-4225	
			106-4226	106-4227	106-4228	106-4229	106-4230					
\$TIMES		001212	#21-1496	*47-2450	*92-4097	92-4097	*92-4097	92-4097	92-4097	92-4097	*95-4103	
\$TKB		001144	#21-1496	97-4107	97-4107	97-4107	97-4107	100-4113	100-4113	100-4113	100-4113	100-4113
\$TKS		001142	#21-1496	54-2644	97-4107	97-4107	97-4107	97-4107	100-4113	100-4113	100-4113	100-4113
			100-4113									
\$TMPO		001174	#21-1496	*38-2149	38-2173	*56-2820	*56-2825	*57-2881	*57-2886	57-2887	57-2904	
			*61-3048	*63-3132	64-3157	64-3163	64-3186	65-3201	65-3220	*68-3286	*68-3352	
			68-3370	*68-3376	*79-3754	*79-3757	*79-3758	*83-3853	83-3854	83-3856	*83-3860	
			83-3861	*84-3879	84-3880	*84-3882	84-3883	*84-3887	84-3888	*85-3906	85-3907	
			85-3909	*85-3913	85-3914	106-4217	106-4218	106-4222	106-4223	106-4230		

SYMBOL	CROSS REFERENCE VALUE	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES	REFERENCES
\$TMP1	001176	#21-1496	*42-2286	*42-2289	*44-2369	*48-2465	*49-2480	*61-3049	106-4223	106-4230
\$TMP2	001200	#21-1496	*42-2290	*44-2368	*45-2403	*45-2404	*45-2405	45-2406	*45-2406	*59-2953
		59-2998	*83-3854	*83-3855	83-3856	83-3861	*84-3880	*84-3881	84-3883	84-3888
		*85-3907	*85-3908	85-3909	85-3914					
\$TMP3	001202	#21-1496	*57-2894	57-2895	*68-3329	68-3330	106-4221			
\$TMP4	001204	#21-1496	*88-4036	*88-4041	106-4225					
\$TMP5	001206	#21-1496	*41-2227	41-2232	41-2233	*60-3028	*60-3030	61-3070	61-3073	*88-4042
		106-4219	106-4225							
\$TMP6	001210	#21-1496	*60-3015	*60-3023	60-3028					
\$TN	= 000032	#16-1471	16-1485	54-2753	55-2753	55-2753	55-2753	#55-2753	55-2790	55-2808
		56-2808	56-2808	56-2808	#56-2808	56-2851	56-2853	56-2876	57-2876	57-2876
		57-2876	#57-2876	58-2952	59-2952	59-2952	59-2952	#59-2952	59-3011	60-3011
		60-3011	60-3011	#60-3011	61-3071	61-3087	62-3087	#62-3087	67-3280	68-3280
		68-3280	68-3280	#68-3280	68-3399	68-3411	69-3411	69-3411	69-3411	#69-3411
		69-3428	69-3460	69-3472	70-3472	70-3472	70-3472	#70-3472	70-3489	71-3489
		71-3489	71-3489	#71-3489	71-3503	72-3503	#72-3503	73-3567	74-3567	74-3567
		74-3567	#74-3567	74-3584	75-3584	75-3584	75-3584	#75-3584	75-3598	76-3598
		76-3598	76-3598	#76-3598	76-3633	77-3633	77-3633	77-3633	#77-3633	77-3645
		77-3648	77-3654	77-3689	78-3689	78-3689	78-3689	#78-3689	78-3711	78-3724
		79-3724	79-3724	79-3724	#79-3724	79-3778	80-3778	80-3778	80-3778	#80-3778
		80-3783	80-3796	81-3796	81-3796	81-3796	#81-3796	81-3806	81-3833	82-3833
		82-3833	82-3833	#82-3833	82-3836	82-3846	83-3846	83-3846	83-3846	#83-3846
		83-3873	84-3873	84-3873	84-3873	#84-3873	84-3900	85-3900	85-3900	85-3900
		#85-3900	86-3972	87-3972	87-3972	87-3972	#87-3972	87-4001	87-4015	88-4015
		88-4015	88-4015	#88-4015						
\$TPB	001150	#21-1496	97-4107	97-4107	97-4107					
\$TPFLG	001155	#21-1496	97-4107	97-4107	97-4107					
\$TPS	001146	#21-1496	97-4107	97-4107	97-4107					
\$TRAP	023520	54-2655	#101-4115							
\$TRAP2	023542	#101-4115	101-4115							
\$TRP	= 000016	#101-4115	101-4115	101-4115	101-4115	101-4115	#101-4115	101-4115	101-4115	101-4115
		101-4115	#101-4115	101-4115	101-4115	101-4115	101-4115	#101-4115	101-4115	101-4115
		101-4115	101-4115	#101-4115	101-4115	101-4115	101-4115	101-4115	#101-4115	101-4115
		101-4115	101-4115	101-4115	#101-4115	101-4115	101-4115	101-4115	101-4115	#101-4115
		101-4115	101-4115	101-4115	101-4115	#101-4115	101-4115	101-4115	101-4115	101-4115
		#101-4115	101-4115	101-4115	101-4115	101-4115	#101-4115	101-4115	101-4115	101-4115
		101-4116	#101-4116	101-4117	101-4117	101-4117	101-4117	#101-4117	101-4118	101-4118
		101-4118	101-4118	#101-4118						
\$TRPAD	023554	101-4115	#101-4115							
\$STSM	020004	#89-4067								
\$STSNM	001100	#21-1496	*47-2446	47-2447	*73-3524	*73-3528	*77-3640	*78-3710	*87-3982	92-4097
		92-4097	*92-4097	92-4097	92-4097	92-4097	92-4097	93-4099	93-4099	93-4099
		93-4099	*95-4103							
\$TTYIN	023454	100-4113	100-4113	100-4113	#100-4113					
\$TYPBN	= *****	101-4115								
\$TYPDS	022772	#99-4111	101-4115	101-4115						
\$TYPE	022200	94-4101	#97-4107	101-4115	101-4115					
\$TYPEC	022412	97-4107	97-4107	97-4107	#97-4107					
\$TYPEX	022542	97-4107	97-4107	97-4107	#97-4107					
\$TYPOC	022570	#98-4109	101-4115	101-4115						
\$TYPON	022604	98-4109	#98-4109	101-4115						
\$TYPOS	022544	#98-4109	101-4115							

SYMBOL	CROSS REFERENCE	REFERENCES			
SYMBOL	VALUE				
\$UNIT	020154	#91-4095			
\$UNITM	020010	#89-4067			
\$USWR	020166	#91-4095			
\$VECT1	020212	#91-4095			
\$VECT2	020214	#91-4095			
\$XOFF	= 000023	97-4107	97-4107		
\$XON	= 000021	97-4107	97-4107	97-4107	100-4113
\$XTSTR	020372	#92-4097			
\$SGET4	= 000001	#95-4103	#95-4103	95-4103	
\$OFILL	022767	*98-4109	*98-4109	98-4109	#98-4109
\$4OCAT	= *****	92-4097	93-4099		
.\$ASTA	= *****	94-4101	94-4101		
.\$X	= 020000	#89-4067	89-4067		
.\$Y	= 002026	#28-1731	28-1734		

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
COMMEN	#17-1489									
DONE	#16-1346	95-4103								
ENDCOM	#17-1489									
ESCAPE	#17-1489									
GETPRI	#17-1489	#95-4103								
GETSWR	#17-1489									
MSG	#80-3787	#81-3796	#81-3828	#82-3833	#82-3840	#83-3846	#83-3867	#84-3873	#84-3894	#85-3900
	#87-4009	#88-4015								
MSG1	#54-2744	55-2753								
MSG11	#67-3262	68-3280								
MSG12	#68-3403	#69-3411								
MSG13	#69-3464	70-3472								
MSG14	#70-3479	71-3489								
MSG15	#71-3495	#72-3503								
MSG16	#73-3556	#74-3567								
MSG17	#74-3571	75-3584								
MSG2	#55-2794	#56-2808								
MSG20	#75-3590	76-3598								
MSG21	#76-3603	#77-3633								
MSG22	#77-3656	78-3689								
MSG220	#78-3712	79-3724								
MSG221	#79-3772	80-3778								
MSG23	#86-3961	#87-3972								
MSG4	#56-2864	#57-2876								
MSG5	#58-2942	#59-2952								
MSG6	#59-2999	#60-3011								
MSG7	#61-3074	#62-3087								
MULT	#17-1489									
NEWTST	#16-1462	#17-1489	#54-2753	#55-2808	#56-2876	#58-2952	#59-3011	#61-3087	#67-3280	#68-3411
	#69-3472	#70-3489	#71-3503	#73-3567	#74-3584	#75-3598	#76-3633	#77-3689	#78-3724	#79-3778
	#80-3796	#81-3833	#82-3846	#83-3873	#84-3900	#86-3972	#87-4015			
POP	#17-1489	#94-4101	#94-4101	#96-4105	#99-4111	#102-4120	#102-4120			
PUSH	#17-1489	94-4101	94-4101	94-4101	96-4105	99-4111	102-4120	102-4120		
REPORT	#17-1489									
SAVTST	#16-1384	#93-4099								
SETPRI	#17-1489									
SETTRA	#101-4115	101-4115	101-4115	101-4115	101-4115	101-4115	101-4115	101-4115	101-4115	101-4115
	101-4115	101-4116	101-4117	101-4118						
SETUP	#17-1489									
SKIP	#17-1489	#55-2790	#56-2851	#56-2853	#61-3071	#68-3399	#69-3428	#69-3460	#77-3645	#77-3648
	#77-3654	#80-3783	#81-3806	#82-3836	#87-4001					
SKIPJ	#16-1337	78-3711								
SLASH	#17-1489									
SPACE	#16-1343	#17-1489								
SSCOPE	#16-1362	#92-4097								
STARS	#17-1489	19-1494	21-1496	21-1496	29-1736	29-1776	30-1903	30-1913	31-1941	31-1946
	32-1963	32-1964	32-1966	32-1967	32-1969	32-1976	33-1984	33-1992	34-1998	34-2007
	35-2019	35-2026	36-2065	36-2067	36-2069	36-2081	37-2113	37-2120	38-2141	38-2148
	39-2177	39-2184	40-2191	40-2198	41-2205	41-2219	42-2255	42-2265	44-2321	44-2328
	46-2410	46-2424	47-2442	48-2453	48-2454	49-2470	50-2485	50-2546	51-2557	55-2753
	55-2753	56-2808	56-2808	57-2876	57-2876	59-2952	59-2952	60-3011	60-3011	62-3087
	62-3087	63-3107	63-3117	64-3152	64-3156	65-3192	65-3198	66-3223	67-3247	67-3252

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES
.\$EOP	#16-475 95-4103
.\$ERRO	#16-863 #93-4099
.\$POWE	#16-1462 #102-4120
.\$READ	#16-1461 100-4113
.\$SAVE	#16-1461 96-4105
.\$SCOP	#16-1042 92-4097
.\$STRAP	#16-1462 #101-4115
.\$TYPD	#16-1461 #99-4111
.\$TYPE	#16-1461 97-4107
.\$TYPO	#16-1461 98-4109
.1170	#16-1459 17-1489