

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

.REM E

IDENTIFICATION

PRODUCT CODE: AC-T587B-MC
PRODUCT NAME: CVMSBBO 0-2M QUICK VERIFY
PRODUCT DATE: APRIL 1984
MAINTAINER: SMALL SYSTEMS DIAGNOSTIC ENGINEERING

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

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

COPYRIGHT (C): 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	POP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131

1.0 ABSTRACT

THIS PROGRAM WAS CREATED TO DO A QUICK VERIFY TEST OF Q-BUS MEMORY. THIS WAS NEEDED ESPECIALLY FOR THE APPLICATION OF TESTING THE MEMORY OF A MICRO PDP-11 UNDER UFD MODE. WHEN RUN UNDER REGULAR XXDP, THE PROGRAM IS DESIGNED TO GIVE THE USER OF THIS PROGRAM FRIENDLY MESSAGES. THIS PROGRAM CAN BE RUN IN XXDP, STANDALONE MODE, USER FRIENDLY DIAGNOSTIC MODE BUT NOT UNDER THE DIAGNOSTIC SUPERVISOR.

2.0 REQUIREMENTS

2.1 HARDWARE

THIS PROGRAM ASSUMES THAT THE FOLLOWING ARE IN PROPER WORKING CONDITIONS.

1. Q-BUS CPU
2. XXDP, LOAD MEDIA

THERE MUST BE MORE THAN 64K OF MEMORY FOR THIS DIAGNOSTIC TO TEST OUT THE MEMORY.

2.2 SOFTWARE

AFTER LOADING, THE PROGRAM MUST BE STARTED AT LOCATION 200

3.0 PROGRAM DESCRIPTION

THIS PROGRAM IS A QUICK VERIFY TEST ON 0.2M OF Q-BUS MEMORY. IT USES THE MOVING INVERSIONS ALGORITHM ON 12 BITS OF A MEMORY WORD. IT ALSO TESTS OUT THE PARITY ERROR DETECT LOGIC OF THE MEMORY. ALL MESSAGES ARE INTENDED TO BE USER FRIENDLY.

WHEN THE PROGRAM IS RUN THE FOLLOWING IS TYPED OUT.

MEMORY: XXXXK BYTES. TEST TIME YY MINUTES ZZ SECONDS

WHERE:

XXXX - IS THE AMOUNT OF MEMORY IN K
YY - IS THE TIME TO TEST TO THE MINUTE
ZZ - IS THE TIME TO TEST TO THE SECOND

FOR EXAMPLE:

MEMORY: 256K BYTES. TEST TIME 3 MINUTES 25 SECONDS

256K BYTES WILL BE TESTED
IT WILL TAKE 3 MINUTES AND 25 SECONDS TO TEST.

132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151

4.0 ERROR REPORTING

ALL ERROR REPORTING IS DESIGNED TO BE EASILY READ AND UNDERSTOOD.

WHEN AN ERROR OCCURS A "CONTROL E" CAN BE INPUT TO RECEIVE EXPANDED ERROR INFORMATION.

E

.TITLE CVMSBBO 0-2M QUICK VERIFY
.SBTTL HEADER

.ENABL LC
.ENABL ABS

;ENABLE LOWER CASE.
;And absolute locations.

000000

EQUATES

153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193

.SBTTL EQUATES

;Memory Management Equates
;

;***
;Status Register addresses
;---

177572
172516
MMUSR0= 177572
MMUSR3= 172516

;***
;Kernal Mode PDR addresses
;---

172300
172302
172304
172306
172310
172312
172314
172316
KPDR0= 172300
KPDR1= 172302
KPDR2= 172304
KPDR3= 172306
KPDR4= 172310
KPDR5= 172312
KPDR6= 172314
KPDR7= 172316

;***
;Kernal Mode PAR addresses
;---

172340
172342
172344
172346
172350
172352
172354
172356
KPAR0= 172340
KPAR1= 172342
KPAR2= 172344
KPAR3= 172346
KPAR4= 172350
KPAR5= 172352
KPAR6= 172354
KPAR7= 172356

EQUATES

```

195
196
197
198
199
200      004406      RW256= 4406      ;PDR setting for Read/Write, 320 words.
201      077406      RW4096= 77406     ;PDR setting for Read/Write, 4096 words.
202      077402      RO4096= 77402     ;PDR setting for Read/Only, 4096 words.
203      077400      NR4096= 77400     ;PDR setting for non-resident.
204
205      000000      PAR0K= 0          ;PAR setting for a base address of 0.
206      000200      PAR4K= 200        ;PAR setting for a base address of 4K.
207      000400      PAR8K= 400        ;PAR setting for a base address of 8K.
208      000600      PAR12K= 600       ;PAR setting for a base address of 12K.
209      001000      PAR16K= 1000      ;PAR setting for a base address of 16K.
210      001200      PAR20K= 1200     ;PAR setting for a base address of 20K.
211      001400      PAR24K= 1400     ;PAR setting for a base address of 24K.
212      177600      PARIO= 177600    ;PAR setting for the I/O page.
213
214      000001      MMUENA= 1          ;SR0 setting to enable memory management.
215      000020      MMU22A= 20        ;SR3 setting to enable 22 bit mapping.
216
217      ;*****
218      ;Processor Status Word and Memory time out vector
219      ;
220
221      177776      PSW= 177776
222      062400      TIMSCA= 62400
223      000004      TIMEOUT= 4
224
225      ;*****
226      ;Write Package Equates
227      ;
228
229      000015      CR= 15             ;ASCII Return.
230      000012      LF= 12            ;ASCII Line Feed.
231      000021      XON= 21           ;ASCII XON.
232      000023      XOFF= 23          ;ASCII XOFF.
233      000033      ESC= 33           ;ASCII Escape character.
234      000007      BELL= 7           ;ASCII Bell character.
235      177560      DFCON= 177560    ;Default console CSR address.
236
237      ;*****
238      ;EQUATES that allow access to the stack frame. These are the offsets into
239      ;the stack for registers saved during the "Preserve Registers" routine.
240
241      000014      R5SLOT= 14         ;Offset for R5.
242      000012      R4SLOT= 12         ;Offset for R4.
243      000010      R3SLOT= 10         ;Offset for R3.
244      000006      R2SLOT= 6          ;Offset for R2.
245      000004      R1SLOT= 4          ;Offset for R1.
246      000002      R0SLOT= 2          ;Offset for R0.

```

PROGRAM MACROS

```
248          .SBTTL PROGRAM MACROS
249
250          ;***
251          ;The "PRESERVE" macro facilitates calling the standard
252          ;register preservation routine (PREG05).
253          ;
254          .macro PRESERVE
255          JSR    R5,PREG05
256          .endm PRESERVE
257
258
259          ;***
260          ;The "PUSH" macro facilitates pushing information on the stack. Up to
261          ;six items may be placed on the stack with one macro.
262          ;
263          .macro PUSH    A,B,C,D,E,F
264          .if nb A
265          MOV    A,-(SP)
266          .endc
267          .if nb B
268          MOV    B,-(SP)
269          .endc
270          .if nb C
271          MOV    C,-(SP)
272          .endc
273          .if nb D
274          MOV    D,-(SP)
275          .endc
276          .if nb E
277          MOV    E,-(SP)
278          .endc
279          .if nb F
280          MOV    F,-(SP)
281          .endc
282          .endm PUSH
```

PROGRAM MACROS

```
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
```

```

;The "POP" macro facilitates retrieving items from the stack. Up to
;six items may be retrieved with one macro.
;---
;macro POP      F,E,D,C,B,A
;if nb F
MOV      (SP)+,F
;endc
;if nb E
MOV      (SP)+,E
;endc
;if nb D
MOV      (SP)+,D
;endc
;if nb C
MOV      (SP)+,C
;endc
;if nb B
MOV      (SP)+,B
;endc
;if nb A
MOV      (SP)+,A
;endc
;endm  POP

;---
;The "CALL" macro facilitates calling a subroutine with or without
;passing parameters. If parameters are passed into the subroutine,
;the first is passed in R5, the second in R4, etc.
;---
;macro CALL     S,A,B,C,D,E,F
;if nb A
MOV      A,R5
;endc
;if nb B
MOV      B,R4
;endc
;if nb C
MOV      C,R3
;endc
;if nb D
MOV      D,R2
;endc
;if nb E
MOV      E,R1
;endc
;if nb F
MOV      F,R0
;endc
JSR      PC,S
;endm  CALL
```

PROGRAM MACROS

```
337 ;+++
338 ;The "RETURN" macro facilitates returning from a subroutine.
339 ;If parameters are passed back, the "PRESERVE" macro MUST be
340 ;used at the start of the subroutine to set up the stack frame.
341 ;Up to six parameters may be passed back. The first is passed
342 ;in R0, the second in R1 etc.
343 ;---
344 .macro RETURN A,B,C,D,E,F
345 .if nb A
346 MOV A,R0SLOT(SP)
347 .endc
348 .if nb B
349 MOV B,R1SLOT(SP)
350 .endc
351 .if nb C
352 MOV C,R2SLOT(SP)
353 .endc
354 .if nb D
355 MOV D,R3SLOT(SP)
356 .endc
357 .if nb E
358 MOV E,R4SLOT(SP)
359 .endc
360 .if nb F
361 MOV F,R5SLOT(SP)
362 .endc
363 RTS PC
364 .endm RETURN
```

STARTUP

```

366          .SBTTL  Startup
367
368          000200          . =      200
369
370 000200  000167  000774      JMP    MAIN
371
372
373          001000          . =      1000

```

```

;This enables XKDP* to start the program
;and disables loading the supervisor.

```

PTYTBL (DATA SECTION) PARITY CSR TABLE

```

375 .SBTTL PTYTBL (Data section) Parity CSR table
376
377 ;The parity CSR table is located here in order to get it into the
378 ;Read/Write section.
379
380 001000 PTYTBL: .BLKW 33. ;Maximum of 16 CSRs, two words per CSR
381 ;plus one word for a terminator.
382
383 001102 000000 CHBUF: .WORD 0 ;Will contain any character received by KBDINT.
384
385 001104 000000 UFDCHR: .WORD 0 ;SAVE FOR INPUT CHARA ;UFD 02/08/84
386 001106 000000 GETOUT: .WORD 0 ;GET OUT FLAG ;UFD 02/08/84
387 001110 000000 CCFLAG: .WORD 0 ;CONTROL C FLAG ;UFD 02/08/84
388
389 ;***
390 ;This is the end of the section that is Read/Write during testing.
391 ;***
392
393 001200 . = 1200
394

```

MAIN - 11/23 PLUS - MEMORY TEST, MAIN FLOW

```

396 .SBTTL MAIN - 11/23 Plus - Memory Test, main flow
397 ;*****
398 ;INPUTS: [II1] Processor is in a Reset state.
399 ; [II2] - From CTRLX only, SWSTAC (in XXDP+) is altered.
400 ; [II3] - Absolute location 42 has XXDP+ return/Chain Mode flag.
401 ;
402 ;OUTPUTS: [OO1] - The XXDP+ return/Chain Mode flag in XXRTN.
403 ; [OO2] - The address of SWSTAC (in XXDP+) in SWADD.
404 ;
405 ;CALLING SEQUENCE: [EN1] - Standard entry from XXDP+.
406 ; [*EN2] - JMP MAIN2
407 ; [*EN3] - Standard restart from XXDP+.
408 ;
409 ;SUBORDINATE ROUTINES CALLED: [CL1]-EMT 43 (to XXDP+)
410 ; [CL2]-WRITE [CL3]-TEST [CL4]-WRITLN.
411 ;*****
412
413 001200
414 001200 000240
415 001202 005767 004724
416 001206 001074
417 001210 010667 004716
418 001214 013767 000042 004672
419 001222 013767 000030 004674
420 001230 013767 000032 004670
421 001236 013767 000052 004664
422 001244 032737 000040 000052
423 001252 001415
424
425
426
427
428
429 001254 104042
430 001256 005060 000042
431 001262 012767 177777 004630
432 001270 032767 000100 004632
433 001276 001403
434 001300 012767 177777 004614
435
436 001306 105737 000052 3$:
437 001312 100002
438 001314 104043
439 001316 000403
440 001320 104042 1$:
441 001322 162700 001424
442 001326 010057 004560 2$:
443
444 001332
445 001332 004767 000546
446 001336 010037 001104
447 001342 022700 000003
448 001346 001006
449 001350 005237 001110
450 001354 005237 001106 2$:
451 001360 004767 000702

```

```

START:
MAIN:  NOP
      TST SAVSP ;FIRST TIME THROUGH? ;UFD 02/09/84
      BNE MAIN1 ;NO SO BRANCH ;UFD 02/09/84
      MOV SP,SAVSP ;SAVE STACK POINTER ;UFD 02/09/84
      MOV @#42,XXRTN ;Save the XXDP+ return/Chain Mode flag in XXRTN
      MOV @#30,SAV30 ;Save the EMT vector
      MOV @#32,SAV32 ;
      MOV @#52,SAV52 ;SAVE LOC 52 ;UFD 02/09/84
      BIT @#40,@#52 ;Are we under UFD CHAIN MODE?
      BEQ 3$ ;No, then continue testing

;+
; IF WE GET HERE WE ARE IN UFD CHAIN MODE
;-
      EMT 42 ;Get DSRERR address
      CLR 42(R0) ;Initialize DSRERR to no error
      MOV @-1,UFDCHN ;Flag that we are under UFD CHAIN MODE
      BIT @#100,SAV52 ;IN QUIET MODE? ;UFD 02/09/84
      BEQ 3$ ;BR IF NOT ;UFD 02/09/84
      MOV @-1,UFDQUI ;SET QUIET FLAG ;UFD 02/09/84

3$: TSTB @#52 ;IS THIS THE UFD MONITOR ? ;UFD 04/03/84
    BPL 1$ ;BR IF NOT ;UFD 04/03/84
    EMT +43 ;Get the address of SWSTAC from XXDP+.
    BR 2$ ;Go save it
1$: EMT +4 ;Get the address of dca from monitor
    SUB @#1424,R0 ;Adjust it to SWSTAC ;UFD 02/09/84
    MOV R0,SWADD ;Save it in SWADD.

CHKCC: CALL GETCHR ;SEE IF A CHARA BEEN TYPED ;UFD 02/09/84
       JSR PC,GETCHR
       MOV R0,@#UFDCHR
       CMP @#3,R0 ;IS IT CONTROL C ? ;UFD 02/09/84
       BNE 1$ ;NO SO BRANCH ;UFD 02/09/84
       INC @#CCFLAG ;SET CC FLAG ;UFD 02/09/84
2$: INC @#GETOUT ;GET GET OUT FLAG ;UFD 02/09/84
    CALL CTRLX ;YES ;UFD 02/09/84
    JSR PC,CTRLX

```

MAIN - 11/23 PLUS - MEMORY TEST, MAIN FLOW

```

451 001364 005767 004550      1$:   TST      UFDCHN      ;IN UFD CHAIN MODE?      ;UFD 02/09/84
452 001370 001403              BEQ      MAIN1        ;NO SO BRANCH           ;UFD 02/09/84
453 001372 022700 000032      CMP      *32,R0       ;IS IT CONTROL Z ?     ;UFD 02/09/84
454 001376 001766              BEQ      2$           ;YES                     ;UFD 02/09/84
455
456
457
458      ;--
459      ; WE MUST TEST FOR AN ORION PROCESSOR. IF WE HAVE ONE THEN TURN OFF CACHE.
460      ;--
461 001400      MAIN1:
462 001400 013746 000010      MOV      @*10,(SP)    ;SAVE LOC 10            ;UFD 04/03/84
463 001404 012737 001446 000010      MOV      *100$,@*10  ;SET FOR NONEX INST TRAP ;UFD 04/03/84
464 001412 000007              MFPT              ;GET PROCESSOR TYPE    ;UFD 04/03/84
465 001414 022700 000005      CMP      *5,R0       ;IS IT AN ORION ?     ;UFD 04/03/84
466 001420 001017              BNE      101$        ;BR IF NOT             ;UFD 04/03/84
467 001422 012767 177777 004506      MOV      *-1,ORIONF   ;SET FLAG              ;UFD 04/03/84
468 001430 013767 177746 004476      MOV      @*177746,SAVCAC ;SAVE CACHE            ;UFD 04/03/84
469 001436 012737 001000 177746      MOV      *1000,@*177746 ;TURN OFF CACHE       ;UFD 04/03/84
470 001444 000403              BR        101$        ;LEAVE NOW            ;UFD 04/03/84
471
472 001446 012716 001454      100$:  MOV      *101$,(SP)   ;WE TRAPPED SO SET UP RTN ;UFD 04/03/84
473 001452 000002              RTI                ;CLEAR STACK FROM TRAP ;UFD 04/03/84
474
475 001454 012637 000010      101$:  MOV      (SP)+,@*10    ;RESTORE LOC 10        ;UFD 04/03/84
476
477 001460 010706              MOV      PC,SP       ;Initialize the stack pointer.
478 001462 042706 017777              BIC      *17777,SP   ;
479 001466 062706 001000              ADD      *1000,SP    ;
480
481 001472              CALL     WRITE,*MTINPG ;Display the "test in progress" message.
482 001472 012705 000004      MOV      *MTINPG,R5
483 001472 004767 004102      JSR      PC,WRITE
484
485 001502              CALL     TEST
486 001502 004767 000506      JSR      PC,TEST     ;Do all the testing.
487
488 001506              CALL     WRITE,*MTOK
489 001506 012705 000074      MOV      *MTOK,R5   ;Display the "OK" message.
490 001512 004767 004066      JSR      PC,WRITE
491
492 001516 005767 004414      MAIN2:  TST      ORIONF      ;ORION CP ?            ;UFD 04/03/84
493 001522 001403              BEQ      9$          ;BR IF NOT             ;UFD 04/03/84
494 001524 016737 004404 177746      MOV      SAVCAC,@*177746 ;RESTORE CACHE         ;UFD 04/03/84
495 001532      9$:
496 001532 016737 004372 000052      MOV      SAV52,@*52  ;RESTORE LOC 52        ;UFD 02/09/84
497 001540 016737 004360 000030      MOV      SAV30,@*30  ;RESTORE 30            ;UFD 02/09/84
498 001546 016737 004354 000032      MOV      SAV3,@*32   ;RESTORE 32            ;UFD 02/09/84
499 001554 016737 004334 000042      MOV      XXRTN,@*42  ;Restore location 4.
500 001562 001415              BEQ      2$          ;If not in chain mode, loop.
501
502 001564 005737 001110      TST      @*CCFLAG    ;PC TYPED ?           ;UFD 02/09/84
503 001570 001407              BEQ      1$          ;NO, SO BRANCH        ;UFD 02/09/84
504 001572 105737 000052      TSTB     @*52        ;IS THIS THE UFD MONITOR ? ;UFD 04/03/84
505 001576 100004              BPL      1$          ;BR IF NOT             ;UFD 04/03/84
506 001600 104042              EMT      4$          ;GET ADDR OF DCA      ;UFD 02/09/84
507 001602 012760 000001 000046      MOV      *1,46(R0)   ;SET PAUS CNT TO 1    ;UFD 02/09/84

```

B?

MAIN 11/23 PLUS - MEMORY TEST, MAIN FLOW

```

503 001610 004777 004300      1$:  JSR    PC, @XXRTN      ;Return to XXDP, CHAIN FILE
504 001614 000406              UR      4$                ;CHAIN RTN                ;UFD 02/09/84
505
506 001616 005737 001106      2$:  TST    @GETOUT      ;GET OUT FLAG SET?      ;UFD 02/09/84
507 001622 001403              BEQ    4$                ;NO SO BRANCH           ;UFD 02/09/84
508 001624 016706 004302      MOV    SAVSP, SP      ;RESTOR STACK PIONTER   ;UFD 02/09/84
509 001630 000207              RTS     PC              ;RETURN TO XXDP, MONITOR ;UFD 02/09/84
510
511 001632 012705 000001      4$:  CALL   WRITE, @NEWLIN ;Do a return and line feed for multiple passes.
      001637 004767 003742      MOV    @NEWLIN, R5
      001636 004767 003742      JSR    PC, WRITE
512 001642 000656              BR     MAIN            ;Loop if multiple Chain Mode passes.

```

KTINT HANDLE MEMORY MANAGEMENT TRAPS

```

514 .SBTTL KTINT - Handle Memory Management traps
515 ;*****
516 ;INPUTS:      [DI1]  TKTERR, the address of the error message table.
517 ;
518 ;OUTPUTS:     [AD01] - KT trap error message to the console.
519 ;
520 ;CALLING SEQUENC:  [*AV2] - Trap through vector 250.
521 ;
522 ;SUBORDINATE ROUTINES CALLED:  [(L1)-ERROR.
523 ;*****
524
525 001644 004736      KTINT: JSR      PC,@(SP)+      ;Routine locator.
526
527 001646            PUSH     R5                ;Save the parameter passing register.
528 001646 010546      MOV      R5,-(SP)
529 001650 004767 005146      JSR      PC,TKTERR      ;Get the address of the error message table.
530 001654            CALL     ERRCR,(SP)+      ;Signal the error using the table address.
531 001654 012605      MOV      (SP)+,R5
532 001656 004767 003562      JSR      PC,ERROR
533
534 001662 012737 000001 177572      MOV     @MMUENA,@MMUSRO ;Reset Memory management.
535 001670            POP      R5                ;Restore the register.
536 001670 012605      MOV      (SP)+,R5
537 001672 000002      RTI                    ;Return from interrupt.

```

PTYINT HANDLE PARITY TRAPS

```

537 .SBTTL PTYINT- Handle Parity traps
538 ;*****
539 ;INPUTS:      [DI1]  -The address of the parity CSR table (PTYTBL).
540 ;
541 ;OUTPUTS:     [DO1]  -The CSR which caused the trap is reset for write
542 ;               good parity, traps enabled.
543 ;               [DO2]  -The error count associated with the CSR is
544 ;                       incremented.
545 ;
546 ;CALLING SEQUENCE:  [*AV2] (Trap through vector 114)
547 ;
548 ;COMMENTS:      If no CSR can be found which caused a trap, exit to UNXINT,
549 ;               to signal an unexpected interrupt.
550 ;
551 ;SUBORDINATE ROUTINES CALLED:  [*CL1] - UNXINT.
552 ;*****
553
554 001674 004736 PTYINT: JSR      PC,@(SP)+      ;Routine locator.
555
556 001676      PUSH     RO              ;Save a working register.
557 001676 010046      MOV      RO,-(SP)
558 001700 012700 001000      MOV      @PTYTBL,RO      ;Address of PTYTBL in the working register.
559 001704 000401      BR       2$
560
561 001706 005720 1$:      TST      (RO)+      ;Get past the error count.
562
563 001710 005710 2$:      TST      (RO)      ;Check for the table terminator.
564 001712 001003      BNE     3$
565 001714      POP      RO              ;If terminator found, restore RO
566 001714 012600      MOV      (SP)+,RO
567 001716 000167 004614      JMP      VCT114      ;And go signal an unexpected trap.
568
568 001722 005730 3$:      TST      @ (RO)+      ;Check the CSR for a parity error.
569 001724 100370      BPL     1$
570
571 001726 005210      INC      (RO)      ;Bump the error count.
572 001730 012750 000001      MOV      @1,@-(RO)      ;Reset the Parity CSR.
573
574 001734      POP      RO              ;Restore the working register.
575 001734 012600      MOV      (SP)+,RO
575 001736 000002      RTI

```

UNXINT - HANDLE UNEXPECTED INTERRUPTS.

```

577 .SBTTL UNXINT- Handle unexpected interrupts.
578 ;*****
579 ;INPUTS:      [DI1]  The address of the vectors table.
580 ;
581 ;OUTPUTS:     [AD01] - Error message to the console.
582 ;
583 ;CALLING SEQUENCE:  [*AV3] Trap through any unused vector.
584 ;
585 ;COMMENTS:     UNXERR does the error handling for this routine.
586 ;
587 ;SUBORDINATE ROUTINES CALLED:  [CL1]-UNXERR.
588 ;*****
589
590 UNXINT; JSR      PC,VCTORS      ;Get the address of the vector table.
591
592          SUB      (SP)+,(SP)    ;
593          SUB      #4,(SP)      ;Form the address of the trapping vector.
594
595          MOV      (SP),-(SP)    ;Form a stack frame and save R5.
596          MOV      R5,2(SP)     ;
597
598          CALL     UNXERR,(SP)+  ;Pass the vector address to the error handler.
599          MOV      (SP)+,R5
600          JSR      PC,UNXERR
601
602          POP      R5           ;Restore R5.
603          MOV      (SP)+,R5
604
605          RTI                ;Return from interrupt.

```

KBDINT - HANDLE CONSOLE KEYBOARD INTERRUPTS.

```

604      .SBTTL KBDINT - Handle console keyboard interrupts.
605      ;*****
606      ;INPUTS:      [DI1]  The input character.
607      ;
608      ;OUTPUTS:     [DO1] - The character is placed in CHBUF.
609      ;
610      ;CALLING SEQUENCE:  [*AV4] - Interrupt through vector 60.
611      ;
612      ;COMMENTS:     If the input character is a Control X, exit the program via
613      ;                CTRLX.
614      ;
615      ;SUBORDINATE ROUTINES CALLED:  [*CL1]-CTRLX.
616      ;*****
617
618 001772 004736      KBDINT: JSR      PC,@(SP)+      ;Routine locator.
619
620 001774 000240      NOP
621 001776 017737 004140 001102      MOV      @*BBUF,@*CHBUF      ;Put the character in CHBUF. ;UFD 02/08/84
622 002004 042737 177600 001102      BIC      @177600,@*CHBUF      ;Strip the character to 7 bits.
623 002012 005767 004104      TST      UFDQUI      ;IS IT UFD QUIET MODE? ;UFD 02/08/84
624 002016 001004      BNE      2$      ;DON'T TEST +X IF IT IS ;UFD 02/08/84
625 002020 022737 000030 001102      CMP      @30,@*CHBUF      ;See if the character is Control X.
626 002026 001420      BEQ      6$      ;IT IS SO GOTO CTRLX ;UFD 02/08/84
627
628 002030 022737 000003 001102 2$:  CMP      @3,@*CHBUF      ;IS IT +C ? ;UFD 02/08/84
629 002036 001410      BEQ      4$      ;YES, SO BRANCH ;UFD 02/08/84
630 002040 005767 004054      TST      UFDCHN      ;UFD CHAIN MODE ? ;UFD 02/08/84
631 002044 001416      BEQ      8$      ;LEAVE NOW IF NOT ;UFD 02/08/84
632 002046 022737 000032 001102      CMP      @3,@*CHBUF      ;IS IT A +Z ? ;UFD 02/08/84
633 002054 001403      BEQ      5$      ;YES, SO LEAVE ;UFD 02/08/84
634 002056 000411      BR      8$      ;IGNORE CHARACTER ;UFD 02/08/84
635
636 002060 005237 001110      4$:  INC      @*CCFLAG      ;SET +C FLAG ;UFD 02/08/84
637 002064 005237 001106      5$:  INC      @*GETOUT      ;SET GET OUT FLAG ;UFD 02/08/84
638 002070 013737 001102 001104 6$:  MOV      @*CHBUF,@*UFDCHR ;SAVE IT ;UFD 02/08/84
639 002076      CALL     CTRLX      ;If it is, Goto CTRLX.
        002076 004767 000164      JSR      PC,CTRLX
640
641 002102 000002      8$:  RTI      ;Return from interrupt.

```

GETCHR - (WRITE PACKAGE) GET A SINGLE CHARACTER INPUT FROM THE

```

643      .SBTTL GETCHR - (Write Package) Get a single character input from the console
644      ;*****
645      ;INPUTS:      [DI1]  CHBUF,
646      ;                [*DI2] - The input character.
647      ;
648      ;OUTPUTS:     [OE1] - R0 contains the character, stripped to 7 bits.  If no
649      ;                character is received, R0 is returned zero.
650      ;
651      ;CALLING SEQUENCE:  CALL  GETCHR
652      ;
653      ;SUBORDINATE ROUTINES CALLED:  None.
654      ;*****
655
656 002104 013700 001102  GETCHR: MOV    @CHBUF,R0    ;Check the character buffer.
657 002110 001403                BEQ    1$                ;
658
659 002112 005037 001102                CLR    @CHBUF          ;If a character was in CHBUF, clear CHBUF.
660 002116 000405                BR     2$                ;And exit.
661
662 002120 105777 004014  1$:   TSTB  @KBCSR          ;Check for inputs.
663 002124 100002                BPL   2$                ;If none, get out.
664
665 002126 117700 004010                MOVB  @KBBUF,R0        ;If input, get it.
666
667 002132 042700 177600  2$:   BIC   @177600,R0      ;Strip to 7 bits.
668
669 002136                RETURN
        002136 000207                RTS    PC

```

PRTCHR - (WRITE PACKAGE) DISPLAY A SINGLE CHARACTER ON THE CON

```

671 .SBTTL PRTCHR - (WRITE PACKAGE) Display a single character on the console
672 ;*****
673 ;INPUTS:      [IE1]  R5 contains the character to be displayed.
674 ;
675 ;OUTPUTS:     [D01] - The character is displayed on the console.
676 ;
677 ;COMMENTS:    If an XOFF is received from the console terminal, this
678 ;              routine waits for an XON before returning.
679 ;
680 ;CALLING SEQUENCE:  CALL  PRTCHR,IE1
681 ;
682 ;SUBORDINATE ROUTINES CALLED:  [CL1],[*CL2]-GETCHR.
683 ;*****
684
685 002140 005767 003756 PRTCHR: TST      UFDQUI      ;Are we running under UFD ?
686 002144 001401      BEQ      FPRTCH      ;Yes,then skip the timeout
687 002146      RETURN      ;UFD 02/08/84
688 002146 000207      RTS      PC
689 002150      FPRTCH: PUSH     R0          ;Save R0.
690 002150 010046      MOV     R0,-(SP)
691 002152 105777 003766 1$:      TSTB     @DSPCSR      ;Check the transmit ready bit on the console.
692 002156 100375      BPL     1$          ;If not ready, wait for it.
693 002160 110577 003762      MOVB     R5,@DSPBUF      ;Send the character.
694
695 002164      CALL    GETCHR      ;Get an input character.
696 002164 004767 177714      JSR     PC,GETCHR
697 002170 022700 000023      CMP     @XOFF,R0        ;See if it is an XOFF.
698 002174 001005      BNE     3$          ;If not, ignore it and get out.
699 002176      2$:
700 002176      CALL    GETCHR      ;If XOFF, wait for an XON.
701 002176 004767 177702      JSR     PC,GETCHR      ;Get a character.
702 002202 022700 000021      CMP     @XON,R0        ;See if it is an XON.
703 002206 001373      BNE     2$          ;If not, loop and wait.
704
705 002210      3$:      POP     R0          ;If XON, return.
706 002210 012600      MOV     (SP)+,R0
707 002212      RETURN
708 002212 000207      RTS     PC

```

TEST TEST ALL OF MAIN MEMORY.

```

708 .SBTTL TEST - Test all of main memory.
709 ;*****
710 ;INPUTS: [II1] The processor is in a Reset state.
711 ;
712 ;OUTPUTS: [ADO1] - Memory is sized. The size and test time is displayed.
713 ; [ADO2] - Memory is tested. Any errors are displayed on the
714 ; console.
715 ;
716 ;CALLING SEQUENCE: CALL TEST
717 ;
718 ;SUBORDINATE ROUTINES CALLED: [CL1]-MAPINI [CL2]-SIZMEM [CL3]-TPARTY
719 ; [CL4]-TL16K [CL5]-TREST [CL6]-CPARTY
720 ;*****
721
722 002214 000240 TEST: NOP ; ;UFD 03/13/84
723 002216 000250 CALL MAPINI ;Set initial conditions for testing.
724 002222 000240 JSR PC,MAPINI
725 002224 000420 NOP ; ;UFD 03/13/84
726 002230 000240 CALL SIZMEM ;Size the memory and get back the highest
727 002232 000750 JSR PC,SIZMEM ;Size the memory and get back the highest
728 002234 000240 MOV RO,R5 ;responding PAR setting. ;UFD 03/13/84
729 002240 000240 JSR PC,TPARTY ;Test parity detect and enable parity traps.
730 002242 001172 CALL TL16K ;Pass the highest PAR setting. ;UFD 03/13/84
731 002246 000240 JSR PC,TL16K ;Test the low 16K words of memory.
732 002250 001212 CALL TREST ; ;UFD 03/13/84
733 002254 000240 JSR PC,TREST ;Test whatever is left.
734 002256 001530 CALL CPARTY ;Pass the highest PAR setting. ;UFD 03/13/84
735 002262 000240 JSR PC,CPARTY ;Check for any parity errors during testing.
735 002264 000207 NOP ; ;UFD 03/13/84
RTS PC ;Return.

```

CTRLX HANDLE A CONTROL X KEYBOARD INPUT

```

737 .SBTTL CTRLX - Handle a Control X keyboard input
738 ;*****
739 ;INPUTS:      [DI1]  SWADD.
740 ;
741 ;OUTPUTS:     [DO1] - SWSTAC (in XXDP+) is altered to disable Chain Mode.
742 ;
743 ;CALLING SEQUENCE:  CALI.  CTRLX
744 ;
745 ;SUBORDINATE ROUTINES CALLED:  [CL1]-SETBAK  [CL2]-MAIN2.
746 ;*****
747
748 002266 013701 001102 CTRLX: MOV      @@CHBUF,R1      ;SAVE LOCATION          ;UFD 02/10/84
749 002272 013702 001104      MOV      @@UFDCHR,R2      ;SAVE LOCATION          ;UFD 02/10/84
750 002276 013703 001106      MOV      @@GETOUT,R3      ;SAVE LOCATION          ;UFD 02/10/84
751 002302 013704 001110      MOV      @@CCFLAG,R4      ;SAVE LOCATION          ;UFD 02/10/84
752
753 002306          CALL     SETBAK          ;Restore original program location and reset.
754 002306 004767 001742      JSR      PC,SETBAK
755
755 002312 010137 001102      MOV      R1,@@CHBUF      ;RESTORE LOCATION       ;UFD 02/10/84
756 002316 010237 001104      MOV      R2,@@UFDCHR     ;RESTORE LOCATION       ;UFD 02/10/84
757 002322 010337 001106      MOV      R3,@@GETOUT     ;RESTORE LOCATION       ;UFD 02/10/84
758 002326 010437 001110      MOV      R4,@@CCFLAG     ;RESTORE LOCATION       ;UFD 02/10/84
759 002332 000240          NOP
760 002334          CALL     FPRTCH,@'↑      ;PRINT UPARROW          ;UFD 02/08/84
761 002334 012705 000136      MOV      @'↑,R5
762 002340 004767 177604      JSR      PC,FPRTCH
763 002344 062737 000100 001104      ADD     @100,@@UFDCHR    ;MAKE CHARA ASCII       ;UFD 02/08/84
764 002352          CALL     FPRTCH,@@UFDCHR ;PRINT CHARACTER        ;UFD 02/08/84
765 002352 013705 001104      MOV      @@UFDCHR,R5
766 002356 004767 177566      JSR      PC,FPRTCH
767
768 002362          CALL     FPRTCH,@40      ;PRINT A SPACE          ;UFD 02/08/84
769 002362 012705 000040      MOV      @40,R5
770 002366 004767 177556      JSR      PC,FPRTCH
771
772 002372 016700 003514      MOV      SWADD,R0        ;Get the address of SWSTAC in XXDP+.
773 002376 005767 003516      TST     UFDCHN          ;UFD CHAIN MODE ?      ;UFD 02/08/84
774 002402 001014          BNE     2$              ;YES, SO BRANCH        ;UFD 02/08/84
775 002404 022737 000130 001104      CMP     @1'0,@@UFDCHR    ;IS IT AN X ?          ;UFD 02/08/84
776 002412 001025          BNE     8$              ;LEAVE IF NOT           ;UFD 02/08/84
777
778 002414 105710          1$:  TSTB    (R0)            ;Check for the stack terminator.
779 002416 001423          BEQ     8$              ;IF found, get out.
780
781 002420 122720 000101      CMPB   @'A,(R0)+        ;Find the auto mode switch.
782 002424 001373          BNE     1$              ;
783 002426 112740 000130      MOVB   @'X,-(R0)        ;Replace it with the exit switch.
784 002432 000415          BR     8$              ;UFD 02/08/84
785
786 002434 005737 001110      2$:  TST     @@CCFLAG      ;IS IT UFD AND +C ?    ;UFD 02/08/84
787 002440 001412          BEQ     8$              ;NO, SO BRANCH         ;UFD 02/08/84
788 002442 105720          3$:  TSTB    (R0)+          ;FIND END OF STACK     ;UFD 02/08/84
789 002444 001376          BNE     3$              ;
790 002446 112760 000057 177777      MOVB   @' /,-1(R0)      ;LOAD SLASH             ;UFD 02/08/84
791 002454 112720 000136      MOVB   @'↑,(R0)+        ;LOAD UPARROW          ;UFD 02/08/84
792 002460 112720 000103      MOVB   @'C,(R0)+        ;LOAD C                 ;UFD 02/08/84
793 002464 105010          CLRB   (R0)            ;SET END OF STACK      ;UFD 02/08/84
    
```

K2

CTRLX HANDLE A CONTROL X KEYBOARD INPUT

787

788 002466 004767 177024 8\$: JSR PC,MAIN2 :Goto MAIN (EN2).

MAPINI - SET INITIAL PROGRAM CONDITIONS

```

790 .SBTTL MAPINI - Set initial program conditions
791 ;*****
792 ;INPUTS:      [II1]  SWADD is set up with the address of SWSTAC in XXDP+.
793 ;
794 ;OUTPUTS:     [+D01] - XXRTN is cleared if XXDP+ is incorrectly located.
795 ;              [D02] - The SP is adjusted for virtual memory 1000 and down.
796 ;              [D03] - Keyboard interrupts are enabled.
797 ;              [D04] - The virtual vector area is loaded with vectors.
798 ;              [AD01] - The program is in 16K to 20K area, the next 4K is
799 ;                      tested.
800 ;              [AD02] - Memory Management is mapped and enabled,
801 ;                      priority zero is set.
802 ;
803 ;CALLING SEQUENCE:  CALL  MAPINI
804 ;
805 ;COMMENTS:      All registers are corrupted by this routine.
806 ;
807 ;SUBORDINATE ROUTINES CALLED:  [+CL1]-ERROR  [CL2]-BUMP  [CL3]-SETABL
808 ;*****
809 002472 016700 003414 MAPINI: MOV SWADD,R0 ;Get the address of SWSTAC (in XXDP+) for
810 002476 042700 007777 BIC #7777,R0 ;Strip unused bits.
811 002502 022700 150000 CMP #150000,R0 ;Check for correct location.
812 002506 001407 BEQ 1$ ;
813 002510 004767 004330 JSR PC,TLOW32 ;If incorrect, get the message table address.
814 002514 CALL ERROR,(SP)+ ;Do the error-handling.
      002514 012605 MOV (SP)+,R5
      002516 004767 002722 JSR PC,ERROR
815 002522 005067 003366 CLR XXRTN ;Disable return to XXDP+.
816
817 002526 1$: CALL BUMP ;Ensure the program is in the 16K area.
      002526 004767 000772 JSR PC,BUMP
818
819 002532 004767 003662 JSR PC,VECTORS ;test the segment address table area.
820 002536 012600 MOV (SP)+,R0 ;Get the address of the vectors table.
821 002540 012701 100000 MOV #100000,R1 ;
822 002544 012702 000100 MOV #64.,R2 ;Set R1 to address the current 4K.
823 ;Set a counter to load the vector area.
824 002550 010021 2$: MOV R0,(R1)+ ;Load a vector.
825 002552 062700 000004 ADD #4,R0 ;Set up the next address.
826 002556 012721 000340 MOV #340,(R1)+ ;
827 002562 077206 SOB R2 ? ;Until all 64 are loaded.
828
829 002564 004767 177054 JSR PC,KTINT ;Get the address of the KI trap handler.
830 002570 012637 100250 MOV (SP)+,#0100250 ;Set up the KI trap vector.
831 002574 004767 177074 JSR PC,PTYINT ;Get the address of the parity trap handler.
832 002600 012637 100114 MOV (SP)+,#0100114 ;Set up the parity trap vector.
833 002604 004767 177162 JSR PC,KBDINT ;Get the address of the keyboard interrupt
834 002610 012637 100060 MOV (SP)+,#0100060 ;handler and set up the keyboard vector.
835
836 002614 004767 003330 JSR PC,TKMAP1 ;Get the address of the initial map.
837 002620 CALL SETABL,(SP)+ ;Go set it up.
      002620 012605 MOV (SP)+,R5
      002622 004767 001520 JSR PC,SETABL
838
839 002626 012767 100000 003262 MOV #100000,RELFLG ;Set RELFLG to reset the SP, don't relocate.
840 002634 042706 160000 BIC #160000,SP ;Adjust the stack pointer.
841 002640 012777 000100 003272 MOV #100,#KBCSR ;Enable keyboard interrupts.

```

M2

MAPINI - SET INITIAL PROGRAM CONDITIONS

842
843 002646
002646 000207

RETURN
RTS PC

NJ

SIZMEM - DETERMINE THE MEMORY SIZE

```

845 .SRTTL SIZMEM - Determine the memory size
846 ;*****
847 ;INPUTS:      [I1]  Memory management is set up and enabled (22 bit).
848 ;
849 ;OUTPUTS:     [OE1] - The highest valid (responding) PAR setting.
850 ;             [ADO1] - The results of the sizing are displayed.
851 ;
852 ;CALLING SEQUENCE:  CALL  SIZMEM
853 ;
854 ;SUBORDINATE ROUTINES CALLED:  [CL1]-DSPSIZ
855 ;*****
856
857 002650 000240 SIZMEM; NOP ;UFD 03/13/84
858 002652 PRESERVE ;Preserve registers.
859 002652 004567 003174 JSR R5,PREG05
860 002656 013746 000004 MOV @#TIMOUT,-(SP) ;Save the current timeout vector.
861 002662 004767 000072 JSR PC,2$ ;Get the address of the local timeout handler.
862 002666 012637 000004 MOV (SP)+,@#TIMOUT ;Put it in the vector location.
863
864 002672 012700 177600 MOV #177600,R0 ;Set R0 to the I/O page PAR setting.
865
866 002676 162700 000200 1$: SUB #200,R0 ;Set R0 to the next lower PAR setting.
867 002702 010037 172347 MOV R0,@#KPAR1 ;Use it to set PAR 1.
868 002706 012737 000000 037776 MOV #0,@#37776 ;Address something in PAR 1 range.
869 002714 001370 BNE 1$ ;If nothing there, try next lower setting.
870
871 002716 012637 000004 MOV (SP)+,@#TIMOUT ;Restore the timeout vector.
872
873 002722 010002 MOV R0,R2 ;Scale the PAR setting into K bytes.
874 002724 062702 000200 ADD #200,R2 ;
875 002730 006202 ASR R2 ;Divide by 16.
876 002732 042702 100000 BIC #100000,R2 ;
877 002736 006202 ASR R2 ;
878 002740 006202 ASR R2 ;
879 002742 006202 ASR R2 ;
880
881 002744 CALL DSPSIZ,R2 ;Go display the memory size and test time.
882 002744 010205 MOV R2,R5
883 002746 004767 000020 JSR PC,DSPSIZ
884
885 002752 RETURN R0 ;Return the highest responding PAR setting.
886 002752 010066 000002 MOV R0,R0SLOT(SP)
887 002756 000207 RTS PC
888
889 002760 004736 2$: JSR PC,@(SP)+ ;Locater for the local handler.
890
891 002762 042766 000004 000002 BIC #4,2(SP) ;Clear the I flag in the saved PSW.
892 002770 000002 RTI ;Return from interrupt.

```

DSPSI2 - DISPLAY MEMORY SIZE AND TEST TIME

```

890 .SBTTL DSPSI2 Display memory size and test time
891 ;*****
892 ;INPUTS: [IE1] Memory size in Kbytes.
893 ;
894 ;OUTPUTS: Size and test time are displayed on the console.
895 ;
896 ;CALLING SEQUENCE: CALL DSPSI2,IE1
897 ;
898 ;SUBORDINATE ROUTINES CALLED: [CL1]-WRIUSN [CL2]-WRITE [CL3]-WRIUSN
899 ; [+CL4]-WRITE [CL5]-WRIUSN [+CL6]-WRITE
900 ; [CL7]-PRCHR
901 ;*****
902
903 DSPSI2: NOP ;UFD 03/13/84
904 PRESERVE ;Preserve registers.
905 002772 000240 JSR R5,PREG05
906 002774 004567 003052 MOV R5,R2 ;Put size in Kbytes in R2.
907 002774 004567 003052 MOV #33.,-(SP) ;Initialize character count on the stack.
908 003006 CALL WRIUSN,R2,#0,#10. ;Display the memory size in K bytes.
909 003006 010205 MOV R2,R5
910 003010 012704 000000 MOV #0,R4
911 003014 012703 000012 MOV #10.,R3
912 003020 004767 002614 JSR PC,WRIUSN
913 003024 060016 ADD R0,(SP) ;Update the character count.
914 003026 012705 000020 CALL WRITE,#MTTIME ;Display the "test time" message.
915 003032 004767 002546 MOV #MTTIME,R5
916 003036 070227 062400 JSR PC,WRITE
917 003042 010203 MUL #TTMSCA,R2 ;Scale K bytes into test time in seconds.
918 003044 005002 MOV R2,R3
919 003046 071227 000074 CLR R2
920 003052 010301 DIV #60.,R2 ;Convert seconds to minutes and seconds.
921 003054 CALL WRIUSN,R2,#0,#10. ;Save seconds.
922 003054 010205 CALL WRIUSN,R2,#0,#10. ;Display test time minutes.
923 003056 012704 000000 MOV R2,R5
924 003062 012703 000012 MOV #0,R4
925 003066 004767 002546 MOV #10.,R3
926 003072 060016 JSR PC,WRIUSN
927 003074 005700 ADD R0,(SP) ;Update the character count.
928 003076 001406 TST R0 ;Check for no minutes.
929 003100 BEQ 1$ ;If no minutes, don't display "minutes".
930 003100 012705 000047 CALL WRITE,#MTTIMM ;Display "minutes".
931 003104 004767 002474 MOV #MTTIMM,R5
932 003110 062716 000017 JSR PC,WRITE
933 003114 CALL WRIUSN,R1,#0,#10. ;Update the character count.
934 003114 010105 1$: CALL WRIUSN,R1,#0,#10. ;Display seconds.
935 003116 012704 000000 MOV R1,R5
936 003122 012703 000012 MOV #0,R4
937 003126 004767 002506 MOV #10.,R3
938 003132 060016 JSR PC,WRIUSN
939 003134 005700 ADD R0,(SP) ;Update the character count.
940 TST R0 ;Check for no seconds.

```

DSPS17 - DISPLAY MEMORY SIZE AND TEST TIME

```

930 003136 001406          BEQ      2#          ;If no seconds, don't display "seconds".
931 003140          CALL     WRITE, #MTAIL ;Display seconds message.
      003140 012705 000062    MOV     #MTAIL, R5
      003144 004767 002434    JSR    PC, WRITE
932 003150 062716 000011    ADD     #9., (SP)          ;Update the character count.
933
934 003154 012600          2#:    MOV     (SP), R0          ;Get the character count.
935 003156 012705 000056    MOV     #'., R5          ;Set up to print dots.
936
937 003162          3#:    CALL    PRTCHR          ;Display a dot.
      003162 004767 176752    JSR    PC, PRTCHR
938 003166 005200          INC     R0              ;Bump the character count.
939 003170 022700 000107    CMP     #71., R0        ;Print until the line is 71 characters.
940 003174 101372          BHI     3#              ;
941
942 003176          CALL    PRTCHR, #40      ;Print a space.
      003176 012705 000040    MOV     #40, R5
      003202 004767 176732    JSR    PC, PRTCHR
943 003206          RETURN
      003206 000207          RTS     PC

```

TPARTY ENABLE PARITY TRAPS AND TEST PARITY DETECT

```

945 .SBTTL TPARTY - Enable parity traps and test parity detect
946 ;*****
947 ;INPUTS: [IE1] The highest valid PAR setting.
948 ; [II1] - Memory Management is set up and enabled (22 bit).
949 ;
950 ;OUTPUTS: [OI1] - PTYTBL, the parity CSR table is established with all
951 ; error counts initialized to zero.
952 ; [OI2] - All live parity CSRs are set for write good parity,
953 ; traps enabled.
954 ;
955 ;CALLING SEQUENCE: CALL TPARTY,IE1
956 ;
957 ;SUBORDINATE ROUTINES CALLED: [CL1]-CPARTY [CL2]-SETABL [CL3]-CPARTY
958 ;*****
959 003210 000240 TPARTY: NOP ;UFD 03/13/84
960 003212 PRESERVE ;Preserve registers.
961 003212 004567 002634 JSR R5,PREG05
962 003216 012704 001000 MOV @PTYTBL,R4 ;Get the address of the parity CSR table.
963 003222 004767 003772 JSR PC,WRNGPA ;Get the address of the "write wrong parity"
964 003226 012605 000422 CALL SPARTY,(SP)+ ;Pass it to SPARTY for setup, set up PTYTBL
965 003226 004767 000422 MOV (SP)+,R5 ;with all error counts at -4.
966 003230 JSR PC,SPARTY ;Save current contents of the parity vector.
967 003234 PUSH @0114
968 003234 013746 000114 MOV @0114,-(SP)
969 003240 004767 000116 JSR PC,7$ ;Get the address of the local trap handler.
970 003244 012637 000114 MOV (SP)+,@0114 ;Set up the parity trap vector.
971 003250 005001 CLR R1 ;Initialize the parity word.
972 003252 005002 CLR R2 ;Initialize the byte/word addressing flag.
973 003254 1$: CALL SETABL ;Set all live CSRs for "write wrong parity".
974 003254 004767 001066 JSR PC,SETABL
975 003260 012703 001600 MOV @1600,R3 ;Set up the first PAR setting.
976 003264 010337 172342 2$: MOV R3,@0KPAR1 ;Set up the working PAR.
977 003270 005702 TST R2
978 003272 100403 BMI 3$
979 003274 110137 037777 MOVB R1,@037777 ;Write something into the PARs range.
980 003300 000402 BR 4$
981 003302 010137 037776 3$: MOV R1,@037776
982 003306 013700 037776 4$: MOV @037776,R0 ;Read it back for possible bad parity.
983 003312 010137 037776 5$: MOV R1,@037776 ;Write good parity.
984 003316 013700 037776 MOV @037776,R0 ;Read back good parity.
985 003322 062703 002000 ADD @2000,R3 ;Bump the PAR setting by 32K words.
986 003326 020366 000016 CMP R3,R5SLOT+2(SP) ;See if the range of memory has been covered.
987 003332 101754 2$ BLOS ;If not, go do it again.
988 003334 005101 COM R1 ;Set R1 for the opposite parity.
989 003336 100746 BMI 1$ ;Do both parities.
990 003340 005702 TST R2
991 003342 100402 BMI 6$ ;If word and byte addressing done, get out.
992 003344 005102 COM R2 ;Set for word addressing.
993 003346 000742 BR 1$ ;loop.

```

TPARTY - ENABLE PARITY TRAPS AND TEST PARITY DETECT

```

997
998 003350          6$:   POP      @0114          ;Restore the parity trap vector.
    003350 012637 000114   MOV      (SP)+,@0114
999 003354          CALL   CPARTY          ;Check that error counts are now all zero.
    003354 004767 000432   JSR      PC,CPARTY
1000 003360         RETURN
    003360 000207         RTS      PC

```

TPARTY ENABLE PARITY TRAPS AND TEST PARITY DETECT

```

1002
1003
1004
1005          ;***
1006          ;Local parity trap handler. Traps with write wrong parity set may
1007          ;result in wrong parity on the stack. This routine cleans the stack,
1008          ;resets the trapping CSR and increments its associated error count.
1009          ;---
1010 003362 004736          7$: JSR    PC,@(SP)+      ;locator for the local handler.
1011
1012 003364 010400          MOV    R4,R0          ;Set R0 to address the parity CSR table.
1013
1014 003366 005710          8$: TST    (R0)          ;Check for end of table.
1015 003370 001005          BNE    9$            ;
1016
1017 003372 004767 003466   JSR    PC,TNOPTY      ;If the program gets here there may be trouble.
1018 003376          CALL   ERROR,(SP)+   ;It means the parity CSR is causing traps but
      003376 012605          MOV    (SP)+,R5
      003400 004767 002040   JSR    PC,ERROR
1019
1020
1021
1022
1023
1024 003404 005770 000000   9$: TST    @0(R0)      ;See if the CSR caused the trap.
1025 003410 100403          BMI    10$          ;
1026 003412 062700 000004   ADD    @4,R0         ;If not, get to the next CSR address.
1027 003416 000763          BR     8$            ;
1028
1029
1030 003420 012730 000001   10$: MOV    @1,@(R0)+   ;If it did, reset the CSR.
1031 003424 005210          INC    (R0)          ;Bump the error count.
1032 003426 010026          MOV    R0,(SP)+     ;Clean up the stack.
1033 003430 010026          MOV    R0,(SP)+     ;
1034 003432 005037 177776   CLR    @0PSW        ;Reset Priority zero.
1035 003436 000705          BR     5$            ;Get back to the test.

```

TL16K - TEST THE LOW 16K WORDS OF MEMORY

```

1037 .SBTTL TL16K - Test the low 16K words of memory
1038 ;*****
1039 ;INPUTS:      [I11]  Memory management is set up and enabled.
1040 ;
1041 ;OUTPUTS:     [A001] - The low 16K words of memory is tested.
1042 ;
1043 ;CALLING SEQUENCE:      CALL   TL16K
1044 ;
1045 ;SUBORDINATE ROUTINES CALLED:  [CL1]-CLRMEM  [CL2]-TSTMEM
1046 ;*****
1047
1048 TL16K:  NOP                               ;UFD 03/13/84
1049        PRESERVE                          ;Preserve registers.
1050        JSR      R5,PREG05
1051        CLR      R5                        ;Set the low PAR setting to 0-4K.
1052        MOV      0600,R4                  ;Set the high PAR setting to 12-16K.
1053
1054        CALL     CLRMEM                   ;Clear the low 16K words.
1055        JSR      PC,CLRMEM
1056        CALL     TSTMEM                   ;Test the low 16K words.
1057        JSR      PC,TSTMEM
1058
1059        RETURN
1060        RTS      PC

```

TREST - TEST THE REMAINDER OF MEMORY

```

1059 .SBTTL TREST - Test the remainder of memory
1060 ;*****
1061 ;INPUTS:      [IE1]  The highest valid PAR setting.
1062 ;             [III]  Memory management is set up and enabled (22 bits).
1063 ;
1064 ;OUTPUTS:     [AD01] - Memory above 16K words is tested.
1065 ;
1066 ;CALLING SEQUENCE:      CALL  TREST,PE1
1067 ;
1068 ;SUBORDINATE ROUTINES CALLED:  [CL1]-SETLOW  [CL2]-CLRMEM  [CL3]-TSTMEM
1069 ;                               [CL4]-SETBAK
1070 ;*****
1071
1072 003466 000240 TREST:  NOP                               ;UFD 03/13/84
1073 003470 PRESERVE                               ;Preserve registers.
1074 003470 004567 002356 JSR    R5,PREG05
1075 003474 010504 MOV    R5,R4                               ;Set up the high PAR setting.
1076 003476 012705 001000 MOV    #1000,R5                             ;Set up the low PAR setting.
1077
1078 003502 CALL  SETLOW                               ;Set up for operation in the low 16K words.
1079 003502 004767 000504 JSR    PC,SETLOW
1080 003506 CALL  CLRMEM                               ;Clear all memory above 16K words.
1081 003506 004767 000336 JSR    PC,CLRMEM
1082 003512 CALL  TSTMEM                               ;Test all that memory.
1083 003512 004767 000406 JSR    PC,TSTMEM
1084 003516 CALL  SETBAK                               ;Restore the program to its old location.
1085 003516 004767 000532 JSR    PC,SETBAK
1086 003522 RETURN
003522 000207 RTS    PC

```

BUMP RELOCATE THE PROGRAM UP UNTIL IT RESIDES IN 16K TO 20

```

1088 .SBTTL BUMP - Relocate the program up until it resides in 16K to 20K
1089 ;*****
1090 ;INPUTS: [DI1] The current program location.
1091 ;
1092 ;OUTPUTS: [*D01] - The program is relocated if not already at 16K.
1093 ; [D02] - The 20K to 24K memory segment is tested (quick test)
1094 ;
1095 ;CALLING SEQUENCE: CALL BUMP
1096 ;
1097 ;COMMENTS: All registers are corrupted by this routine.
1098 ;
1099 ;SUBORDINATE ROUTINES CALLED: [*CL1]-ERROR.
1100 ;*****
1101 003524 010700 BUMP: MOV PC,R0 ;Establish the current program location.
1102 003526 042700 017777 BIC #17777,R0 ;
1103 003532 010002 MOV R0,R2 ;
1104 003534 010001 MOV R0,R1 ;
1105 003536 012705 010000 MOV #4096.,R5 ;Set a 4K counter.
1106 003542 005004 CLR R4 ;Initialize the program checksum.
1107
1108 003544 062204 1$: ADD (R2)+,R4 ;Calculate the program checksum.
1109 003546 077502 SOB R5,1$ ;
1110
1111 003550 012705 010000 2$: MOV #4096.,R5 ;Set a 4K counter.
1112 003554 012012 3$: MOV (R0)+,(R2) ;Copy the program to the next 4K segment.
1113 003556 005112 COM (R2) ;
1114 003560 005122 COM (R2)+ ;Quick-test all locations.
1115 003562 077504 SOB R5,3$ ;
1116
1117 003564 012705 010000 MOV #4096.,R5 ;Set a 4K counter.
1118 003570 005003 CLR R3 ;Initialize the copy checksum.
1119 003572 062103 4$: ADD (R1)+,R3 ;Calculate the copy checksum.
1120 003574 077502 SOB R5,4$ ;
1121
1122 003576 020304 CMP R3,R4 ;Compare the copy checksum to the program
1123 003600 001405 BEQ 5$ ;checksum.
1124 003602 004767 003236 JSR PC,TLOW32 ;If they aren't the same, signal an error.
1125 003606 CALL ERROR,(SP)+ ;
1126 003606 012605 MOV (SP)+,R5 ;
1127 003610 004767 001630 JSR PC,ERROR ;
1128
1127 003614 022707 100000 5$: CMP #100000,PC ;See if the program is high enough.
1128 003620 101415 BLOS 6$ ;
1129 003622 062707 020000 ADD #20000,PC ;If not, jump to the copy in the next 4K.
1130 003626 062706 020000 ADD #20000,SP ;Adjust the stack pointer to the current area.
1131 003632 062716 020000 ADD #20000,(SP) ;Adjust all the returns.
1132 003636 062766 020000 000002 ADD #20000,2(SP) ;
1133 003644 062766 020000 000004 ADD #20000,4(SP) ;
1134 003652 000724 BR BUMP ;Bump again.
1135
1136 003654 6$: RETURN
003654 000207 RTS PC

```

SPARTY - INITIALIZE THE PARITY CSR TABLE AND SET UP WRNGPA

```

1138 .SBTTL SPARTY - Initialize the parity CSR table and set up WRNGPA
1139 ;*****
1140 ;INPUTS:      [IE1]  Address of WRNGPA.
1141 ;             [DI1]  Address of PTYTBL.
1142 ;
1143 ;OUTPUTS:     [DO1]  - PTYTBL is set up with the addresses of all "live"
1144 ;                 parity CSRs and -4 for all error counts.
1145 ;             [DO2]  - WRNGPA is set up with a "write wrong parity" command
1146 ;                 for each "live" parity CSR.
1147 ;
1148 ;CALLING SEQUENCE:  CALL  SPARTY,IE!
1149 ;
1150 ;SUBORDINATE ROUTINES CALLED:  [*CL1]-ERROR
1151 ;*****
1152 003656 000240 SPARTY: NOP ;UFD 03/13/84
1153 003660 PRESERVE ;Preserve registers.
1154 003660 004567 002166 JSR R5,PREG05
1155 003664 012704 001000 MOV @PTYTBL,R4 ;Get the address of PTYTBL.
1156 003670 010403 MOV R4,R3 ;
1157 ;
1158 003672 012702 172100 MOV @172100,R2 ;Set up the address of the parity CSRs.
1159 003676 012701 000020 MOV @16.,R1 ;Set up to check for all CSRs.
1160 ;
1161 003702 PUSH @@TIMOUT ;Save the current time-out handler.
1162 003702 013746 000004 MOV @@TIMOUT,-(SP)
1163 003706 004767 000066 JSR PC,4$ ;Get the address of the local time-out handler.
1164 003712 012637 000004 MOV (SP)+,@@TIMOUT ;Set up the timeout vector.
1165 003716 012712 000000 1$: MOV @0,(R2) ;Test for a CSR.
1166 003722 001006 BNE 2$ ;
1167 ;
1168 003724 012725 000005 MOV @5,(R5)+ ;If the CSR exists, make entr' s in WRNGPA.
1169 003730 010225 MOV R2,(R5)+ ;
1170 003732 010224 MOV R2,(R4)+ ;Make entries in PTYTBL.
1171 003734 012724 177774 MOV @-4,(R4)+ ;
1172 ;
1173 003740 062702 000002 2$: ADD @2,R2 ;Bump to the next CSR address.
1174 003744 005065 000002 CLR 2(R5) ;Ensure a zero word at the end of each table.
1175 003750 005014 CLR (R4) ;
1176 003752 077117 SOB R1,1$ ;Test all 16 possible CSR addresses.
1177 ;
1178 003754 POP @@TIMOUT ;Restore the time-out vector.
1179 003754 012637 000004 MOV (SP)+,@@TIMOUT
1180 003760 020403 CMP R4,R3 ;See if any CSRs were alive.
1181 003762 001005 BNE 3$ ;
1182 003764 004767 003074 JSR PC,INOPTY ;
1183 003770 CALL ERROR,(SP)+ ;If not, signal an error.
1184 003770 012605 MOV (SP)+,R5
1185 003772 004767 001446 JSR PC,ERROR
1186 ;
1187 ;
1188 004000 004736 3$: RETURN
RTS PC
4$: JSR PC,@(SP)+ ;Local timeout handler locator.

```

SPARTY - INITTALIZE THE PARITY CSR TABLE AND SET UF WRNGPA

1189

1190 004002 042766 000004 000002

1191 004010 000002

BIC #4,2(SP)
RTI

;Clear the Z bit in the CSR.
;Return from trap.

CPARTY - CHECK FOR ANY PARITY ERRORS

```

1193 .SBTTL CPARTY - Check for any parity errors
1194 ;*****
1195 ;INPUTS:      [DI1]  The address of PTYTBL.
1196 ;
1197 ;OUTPUTS:     [*AD01] - Error message to the console.
1198 ;
1199 ;CALLING SEQUENCE:  CALL  CPARTY
1200 ;
1201 ;SUBORDINATE ROUTINES CALLED:  [*CL1]-PTYERR
1202 ;*****
1203
1204 004012 000240 CPARTY: NOP ;UFD 03/13/84
1205 004014 004567 002032 PRESERVE ;Preserve registers.
      004014 JSR R5,PREG05
1206
1207 004020 012700 001000 MOV #PTYTBL,R0 ;Get the address of PTYTBL.
1208
1209 004024 005720 1$: TST (R0)+ ;Check for end of table.
1210 004026 001407 BEQ 2$ ;
1211
1212 004030 005720 TST (R0)+ ;Check the error count.
1213 004032 001774 BEQ 1$ ;
1214 004034 016005 177774 CALL PTYERR,-4(R0) ;If error, signal when CSR detected the error.
      004034 004767 000726 MOV -4(R0),R5
1215 004044 000767 JSR PC,PTYERR
      BR 1$ ;
1216
1217 004046 000207 2$: RETURN
      RTS PC

```

CLRMEM - CLEAR ALL SELECTED MEMORY

```

1219 .SBTTL CLRMEM - Clear all selected memory
1220 ;*****
1221 ;INPUTS:      [IE1]  The low PAR setting to be used.
1222 ;             [IE2]  The high PAR setting to be used.
1223 ;             [III]  Memory Management is set up and enabled (22 bits).
1224 ;
1225 ;OUTPUTS:     [ADO1] - All selected memory is set to all zeroes.
1226 ;
1227 ;CALLING SEQUENCE:      CALL  CLRMEM,IE1,IE2
1228 ;
1229 ;SUBORDINATE ROUTINES CALLED:  [CL1]-CLEAR      [*CL2]-SEGERR
1230 ;*****
1231
1232 004050 000240 CLRMEM: NOP ;UFD 03/13/84
1233 004052 PRESERVE ;Preserve registers.
1234 004052 004567 001774 JSR R5,PREG05
1235 004056 010501 MOV R5,R1 ;Move IE1 to a safe register.
1236 004060 010402 MOV R4,R2 ;And IE2.
1237
1238 004062 012705 020000 MOV #20000,R5 ;Set up PE1 for CLEAR.
1239 004066 012704 010000 MOV #4096.,R4 ;And PE2.
1240
1241 004072 010137 172342 1$: MOV R1,#KPAR1 ;Set up the working PAR.
1242 004076 CALL CLEAR ;Clear the 4096 word segment.
1243 004076 004767 000266 JSR PC,CLEAR
1244
1244 004102 005700 TST R0 ;Check the returned test flag.
1245 004104 001402 BEQ 2$ ;
1246 004106 CALL SEGERR ;If error, signal an error in the segment.
1247 004106 004767 001154 JSR PC,SEGERR
1248
1248 004112 062701 000200 2$: ADD #200,R1 ;Bump the PAR setting.
1249 004116 020102 CMP R1,R2 ;See if all selected memory is cleared.
1250 004120 101764 BLOS 1$ ;If not, do the next segment.
1251
1252 004122 RETURN
004122 000207 RTS PC

```

TSTMEM - TEST ALL SELECTED MEMORY

```

1254 .SBTTL TSTMEM - Test all selected memory
1255 ;*****
1256 ;INPUTS:      [IE1]  The low PAR setting.
1257 ;            [IE2]  - The high PAR setting.
1258 ;            [II1]  - Memory Management is set up and enabled (22 bits).
1259 ;
1260 ;OUTPUTS:     [AD01] - All memory selected by IE1 and IE2 is tested.
1261 ;
1262 ;CALLING SEQUENCE:  CALL  TSTMEM,IE1,IE2
1263 ;
1264 ;SUBORDINATE ROUTINES CALLED:  [#CL1]-INVMEM
1265 ;*****
1266
1267 004124 000240 TSTMEM: NOP ;UFD 03/13/84
1268 004126 PRESERVE ;Preserve registers.
1269 004126 004567 001720 JSR R5,PREG05
1270 004132 012703 000001 MOV #1,R3 ;Initialize PE3, tha address increment bit.
1271 004136 005002 CLR R2 ;Initialize the pos/neg flag, PE4.
1272 004140 012700 000014 MOV #12.,R0 ;Initialize the bit shift count.
1273
1274 004144 006303 1$: ASL R3 ;Shift the address increment bit.
1275 004146 012701 010000 MOV #4096.,R1 ;Inverting from zeroes to ones.
1276 004152 CALL INVMEM ;Invert all memory, positive address pattern.
1277 004152 004767 000250 JSR PC,INVMEM
1278 00.156 005102 COM R2 ;Shift to a negative addressing pattern.
1279 004160 005001 CLR R1 ;Inverting from ones to zeroes.
1280 004162 CALL INVMEM ;Invert all memory, negative address pattern.
1281 004162 004767 000240 JSR PC,INVMEM
1282 004166 012701 010000 MOV #4096.,R1 ;Inverting from zeroes to ones.
1283 004172 CALL INVMEM ;Invert all memory, negative address pattern.
1284 004172 004767 000230 JSR PC,INVMEM
1285
1286 004176 005102 COM R2 ;Shift back to a positive addressing pattern.
1287 004200 005001 CLR R1 ;Inverting from ones to zeroes.
1288 004202 CALL INVMEM ;Invert all memory, positive address pattern.
1289 004202 004767 000220 JSR PC,INVMEM
1290
1291 004206 077022 SOB R0,1$ ;test all 12 addressing patterns.
1292
1293 004210 RETURN
1294 004210 000207 RTS PC

```

SETLOW - RELOCATE AND REMAP FOR OPERATION IN THE LOW 16K WORDS

```

1292 .SBTTL SETLOW Relocate and remap for operation in the low 16K words
1293 ;*****
1294 ;INPUTS: [DI1] TKMAP2, the address of the PAR map for low 16K
1295 ; operation.
1296 ;
1297 ;OUTPUTS: [DO1] - RELFLG is set.
1298 ; [ADD1] - The program is relocated to the low 16K words of
1299 ; memory and the KI is remapped for operation there.
1300 ;
1301 ;CALLING SEQUENCE: CALL SETLOW
1302 ;
1303 ;SUBORDINATE ROUTINES CALLED: [CL1]-RLCATE [CL2]-SETABL
1304 ;*****
1305
1306 004212 000240 SETLOW: NOP ;UFD 03/13/84
1307 004214 PRESERVE ;Preserve registers.
1308 004214 004567 001632 JSR R5,PRES05
1309 004220 012737 000000 172342 MOV @PAR0K,@KPAR1 ;Ensure PAR 1 is properly set.
1310
1311 004226 CALL RLCATE ;Relocate the program in physical memory.
1312 004226 004767 000326 JSR PC,RLCATE
1313 004232 JSR PC,TKMAP2 ;Get the address of TKMAP2.
1314 004236 CALL SETABL,(SP); ;Pass it to SETABL.
1315 004236 012605 MOV (SP),R5
1316 004240 004767 000102 JSR PC,SETABL
1317
1318 004244 012767 000001 001644 MOV @1,RELFLG ;Set the relocation flag.
1319
1320 004252 RETURN
1321 004252 000207 RTS PC

```

SETBAK RELOCATE BACK TO ORIGINAL PHYSICAL MEMORY AND RESET

```

1300      ,SBTTL SETBAK - Relocate back to original physical memory and Reset
1301      ;*****
1302      ;INPUTS:      [DI1]  RELFLG.
1303      ;              [DI2]  TKMAP4, the address of the PDR map for all APRs set to
1304      ;              Read/Write.
1305      ;
1306      ;OUTPUTS:    [OI1]  - The processor is in a Reset state.
1307      ;              [OO1]  - RELFLG is cleared.
1308      ;              [ADO1] - The program resides in the 16K to 20K physical
1309      ;              memory area.
1310      ;
1311      ;CALLING SEQUENCE:  CALL  SETBAK
1312      ;
1313      ;SUBORDINATE ROUTINES CALLED:  [+CL1]-SETABL  [+CL2]-RLCATE
1314      ;*****
1315
1316      SETBAK:  NOP                ;UFD 03/13/84
1317      PRESERVE                ;Preserve registers.
1318      JSR      R5,PREG05
1319
1320      1$:  BIT      @77777,RELFLG ;See if a relocation is needed.
1321      BEQ      1$
1322
1323      JSR      PC,TKMAP4          ;Get the address of TKMAP4.
1324      CALL    SETABL,(SP)+       ;Pass it to SETABL.
1325      MOV     (SP)+,R5
1326      JSP     PC,SETABL
1327
1328      MOV     @PAR16K,@@KPAR1    ;Ensure that PAR 1 is properly set.
1329
1330      CLR     RELFLG             ;Reset RELFLG.
1331
1332      CALL    RLCATE             ;Relocate back to original physical memory.
1333      JSR     PC,RLCATE
1334
1335      1$:  RESET
1336
1337      BIS     RELFLG,SP          ;fix the stack pointer.
1338
1339      CLR     R0                 ;Do a timer to allow the Reset to settle.
1340      2$:  NOP
1341      SOB     R0,2$
1342
1343      3$:  NOP
1344      SOB     R0,3$
1345      NOP
1346
1347      ;UFD 03/12/84
1348      ;UFD 03/12/84
1349      ;UFD 03/12/84
1350
1351      RETURN
1352      RTS     PC
1353
1354      004254 000240
1355      004256 004567 001570
1356      004262 032767 077777 001626
1357      004270 001414
1358      004272 004767 002070
1359      004276 012605
1360      004300 004767 000042
1361      004304 012737 001000 172342
1362      004312 005067 001600
1363      004316 004767 000236
1364      004322 000005
1365      004324 056706 001566
1366      004330 005000
1367      004332 000240
1368      004334 077002
1369      004336 000240
1370      004340 077002
1371      004342 000240
1372      004344 000207

```

SETABL SET LOCATIONS FROM A DATA-ADDRESS TABLE

```

1365 .SBTTL SETABL - Set locations from a Data-Address table
1366 ;*****
1367 ;INPUTS:      [IE1]  The address of the table to be used.
1368 ;
1369 ;OUTPUTS:     [OO1] - The locations addressed by the table are set up with
1370 ;                data from the table.
1371 ;
1372 ;CALLING SEQUENCE:  CALL  SETABL,IE1
1373 ;
1374 ;COMMENTS:      The table sequence is:
1375 ;                End      - 0      )
1376 ;                )-      )
1377 ;                )- Address Slot -)
1378 ;                )- Data Slot  -)
1379 ;                )-      )
1380 ;                )-      )
1381 ;                )-      )
1382 ;                )- Address Slot -)
1383 ;                )- Data Slot  -)
1384 ;                )- Address Slot -)
1385 ;                Start -) - Data Slot -)
1386 ;
1387 ;SUBORDINATE ROUTINES CALLED:  None.
1388 ;*****
1389
1390 004346 000240 SETABL: NOP ;D 03/13/84
1391 004350 PRESERVE ;Preserve registers.
1391 004350 0001476 JSR R5,PREG05
1392
1393 004354 012504 1$: MOV (R5)+,R4 ;Get the data.
1394
1395 004356 000115 TST (R5) ;Check for end of table.
1396 004360 001402 BEQ 2$ ;
1397
1398 004362 010430 MOV R4,@(R5)+ ;If not the end, put the data in the location.
1399 004364 000113 BR 1$ ;Continue until end of table.
1400
1401 004366 000207 2$: RETURN
004366 RTS PC

```

CLEAR - CLEAR A SEGMENT OF MEMORY

```

1403 .SBTTL CLEAR - Clear a segment of memory
1404 ;*****
1405 ;INPUTS: [IE1] The starting address of the segment to be cleared.
1406 ; [IE2] The segment size in words.
1407 ;
1408 ;OUTPUTS: [DO1] The segment of memory is cleared.
1409 ; [OE1] zero if no errors are detected, non-zero if any errors
1410 ; are detected.
1411 ;
1412 ;CALLING SEQUENCE: CALL CLEAR IE1,IE2
1413 ;
1414 ;COMMENTS: The selected memory segment is cycled back and forth between
1415 ; all ones and all zeros (to warm it up). This is done with
1416 ; byte addressing.
1417 ;
1418 ;SUBORDINATE ROUTINES CALLED: None.
1419 ;*****
1420
1421 004370 000240 CLEAR: NOP ;UFD 03/13/84
1422 004372 PRESERVE ;Preserve registers.
1423 004372 004567 001454 JSR R5,PREG05
1424 004376 005000 CLR R0 ;Initialize a zero.
1425
1426 004400 110015 1$: MOVB R0,(R5) ;Set the low byte to zero.
1427 004402 105115 COMB (R5) ;Complement it.
1428 004404 105115 COMB (R5) ;Set it back to zero.
1429
1430 004406 112515 MOVB (R5)+,(R5) ;Clear the high byte.
1431 004410 105115 COMB (R5) ;Complement it.
1432 004412 105115 COMB (R5) ;Set it back to zero.
1433
1434 004414 112500 MOVB (R5)+,R0 ;Put it in R0 for transfer to the next byte.
1435
1436 004416 077410 SOB R4,1$ ;Do the complete segment.
1437
1438 004420 RETURN R0 ;Return the error flag [OE1].
004420 010066 000002 MOV R0,ROSL0T(SP)
004424 000207 RTS PC

```

INVMEM - INVERT ALL SELECTED MEMORY

```

1440 .SBTTL INVMEM - Invert all selected memory
1441 ;*****
1442 ;INPUTS: [IE1] The low PAR setting to be used.
1443 ; [IE2] - The high PAR setting to be used.
1444 ; [IE3] - The address increment bit.
1445 ; [IE4] - Zero for positive addressing, non-zero for negative
1446 ; addressing.
1447 ; [IE5] - Zero if inverting from ones to zeroes, 4096 if
1448 ; inverting from zeros to ones.
1449 ; [II1] - Memory Management is set up an enabled (22 bits).
1450 ;
1451 ;OUTPUTS: [AD01] - All selected memory is inverted and checked for
1452 ; errors.
1453 ; [*AD02]- Error messages are displayed on the console terminal.
1454 ;
1455 ;CALLING SEQUENCE: CALL INVMEM,IE1,IE2,IE3,IE4
1456 ;
1457 ;SUBORDINATE ROUTINES CALLED: [*CL1]-SPLADR [CL2]-SMIADR [CL3]-SETABL
1458 ; [*CL4]-INVERT [CL5]-SETABL [*CL6]-SEGERR
1459 ;*****
1460 004426 000240 INVMEM: NOP ;UFD 03/13/84
1461 004430 PRESERVE ;Preserve registers.
1462 004430 004567 001416 JSR R5,PREG05
1463 004434 005702 TST R2 ;Check the pos/neg flag.
1464 004436 001010 BNE 1$ ;
1465 004440 010501 MOV R5,R1 ;If positive, set IE1 as the working setting.
1466 004442 010402 MOV R4,R2 ;Set IE2 as the terminal PAR setting
1467 004444 CALL SPLADR,R3 ;Set up a positive addressing pattern.
1468 004446 004767 000424 MOV R3,R5
1469 004452 012703 000200 JSR PC,SPLADR
1470 004456 000411 BR 3$ ;
1471 004460 010401 1$: MOV R4,R1 ;If negative, set IE2 as the working setting.
1472 004462 010502 MOV R5,R2 ;Set IE1 as the terminal PAR setting.
1473 004464 010405 CALL SMIADR,R3 ;Set up a negative addressing pattern.
1474 004466 004767 000460 MOV R3,R5
1475 004472 012703 177600 JSR PC,SMIADR
1476 004476 000401 BR 3$ ;
1477 004500 000301 2$: ADD R3,R1 ;Update the PAR setting.
1478 004502 016804 000004 3$: MOV R1,SLOT(SP),R4 ;Get the ones/zeroes flag.
1479 004506 004767 001622 JSR PC,TKMAP3 ;Get the address of the PDR Read/Only table.
1480 004512 CALL SETABL,(SP)+ ;Set PDRs 2 through 6 for Read/Only.
1481 004512 012605 MOV (SP)+,R5
1482 004514 004767 177626 JSR PC,SETABL
1483 004520 010407 172342 MOV R1,00KPAR1 ;Set up the working PAR.
1484 004524 CALL INVERT ;Invert a 4K word segment.
1485 004524 004767 000500 JSR PC,INVERT
1486 004530 ADD R4,R0 ;Check for errors.
1487 004532 BEQ 4$ ;
1488 004534 CALL SEGERR ;If error, signal an error in the segment.
1489 004534 004767 000526 JSR PC,SEGERR

```

INVMEM - INVERT ALL SELECTED MEMORY

```

1488
1489 004540 020102          4$:  CMP      R1,R2          ;Check done.
1490 004542 001356          BNE      2$              ;
1491
1492 004544 004787 001616   JSR      PC,TKMAP4      ;Get the address of the Read/Write PDR map.
1493 004550          CALL     SETABL,(SP)+   ;Set all PDRs for Read/Write.
      004550 011603          MOV      (SP)+,R5
      004552 004767 177570   JSR      PC,SETABL
1494 004556          RETURN
      004556 000207          RTS      PC

```

RLCATE - RELOCATE THE PROGRAM

```

1496 .SBTTL RLCATE - Relocate the program
1497 ;*****
1498 ;INPUTS:      None.
1499 ;
1500 ;OUTPUTS:     [D01] - A copy of the contents of memory from 100000 to 157776
1501 ;              is made in memory from 20000 to 77776
1502 ;
1503 ;CALLING SEQUENCE:  CALL  RLCATE
1504 ;
1505 ;SUBORDINATE ROUTINES CALLED:  [+CL1]-ERROR
1506 ;*****
1507
1508 004560 000210 RLCATE: NOP ;UFD 03/13/84
1509 004562 PRESERVE ;Preserve registers.
1510 004566 012700 100000 JSR R5,PREG05
1511 004572 012701 030000 MOV #100000,PC ;Set the base address for the first checksum.
1512 004576 005005 CLR R5 ;Set the word count.
1513 ;Initialize the first checksum.
1514 004600 062005 1$: ADD (R0)+,R5 ;Generate the first checksum.
1515 004602 005505 ADC R5 ;Add the carry bit back in.
1516 004604 077105 SOB R1,1$ ;
1517
1518 004626 012700 100000 MOV #100000,R0 ;Set the base address for the move.
1519 004612 012701 020000 MOV #20000,R1 ;
1520 004616 012702 030000 MOV #12288.,R2 ;Set word count.
1521
1522 004622 012021 2$: MOV (R0)+,(R1)+ ;Move the block.
1523 004624 077202 SOB R2,2$ ;
1524
1525 004626 012700 020000 MOV #20000,R0 ;Set the base address for the second checksum.
1526 004632 012701 030000 MOV #12288.,R1 ;Set word count.
1527 004636 005004 CLR R4 ;Initialize the second checksum.
1528
1529 004640 062004 3$: ADD (R0)+,R4 ;Generate the second checksum.
1530 004642 005504 ADC R4 ;Add the carry bit back in.
1531 004644 077105 SOB R1,3$ ;
1532
1533 004646 010405 CMP R4,R5 ;Check the checksums.
1534 004650 001405 BEQ 4$ ;
1535 004652 004767 002166 JSR PC,TLOW32 ;
1536 004656 CALL ERROR,(SP)+ ;If not a match, signal a bad relocation.
1537 004656 012605 MOV (SP)+,R5 ;
1538 004664 004767 000560 JSR PC,ERROR ;
1539
1537 004664 4$: RETURN
1538 004664 000207 RTS PC

```

UNXERR - SIGNAL AN UNEXPECTED INTERRUPT OR TRAP

```

1540 .SBTTL UNXERR - Signal an unexpected interrupt or trap
1541 ;*****
1542 ;INPUTS:      [IE1]  The address of the vector that trapped
1543 ;
1544 ;OUTPUTS:     [AD01] - Unexpected trap error messages are displayed on the
1545 ;               console.
1546 ;
1547 ;CALLING SEQUENCE:  CALL  UNXERR,IE1
1548 ;
1549 ;SUBORDINATE ROUTINES CALLED:  [CL1]-ERROR      [CL2]-WRITE      [CL3]-WRIUSN
1550 ;                               [CL4]-WRITPG
1551 ;*****
1552
1553 004666 000240 UNXERR: NOP ;UFD 03/13/84
1554 004670 PRESERVE ;Preserve registers.
1555 004670 004567 001156 JSR R5,PREG05
1556 004674 004757 002246 JSR PC,TUNXER ;Get the address of the error message table.
1557 004700 CALL ERROR,(SP)+ ;Signal the error and wait for Control E.
1558 004700 012605 MOV (SP)+,R5
1559 004702 004767 000536 JSR PC,ERROR
1560
1561 004706 CALL WRITE,#ETVCTR ;Display "this is the trapping vector".
1562 004706 012702 001504 MOV #ETVCTR,R5
1563 004712 004767 000566 JSR PC,WRITE
1564
1565 004716 CALL WRIUSN,R5,SLOT(SP),#0,#8. ;Display the trapping vector.
1566 004716 016605 000014 MOV R5,SLOT(SP),R5
1567 004722 012704 000060 MOV #0,R4
1568 004726 012703 000010 MOV #0,R3
1569 004732 004767 000702 JSR PC,WRIUSN
1570
1571 004736 005767 001156 TST R4 IN ;ARE WE IN UFD F.S. MODE? ;UFD 03/08/84
1572 004742 001405 BEQ ;BR IF NOT ;UFD 03/08/84
1573 004744 012737 000032 001104 MOV #32,#0UFDCR ;MAKE LIKE #Z TYPED ;UFD 03/08/84
1574 004752 CALL CTRLX ;LET'S GET OUT ;UFD 03/08/84
1575 004752 004767 175310 JSR PC,CTRLX
1576
1577 004756 15: ;UFD 03/08/84
1578 004756 004757 002176 JSR PC,IGCTU
1579 004762 CALL WRITPG,(SP)+ ;Display "continuing with the test"
1580 004762 012605 MOV (SP)+,R5
1581 004764 004757 000544 JSR PC,WRITPG
1582
1583 004770 RETURN
1584 004770 000207 RTS PC

```

PTYERR - SIGNAL * PARITY DETECT ERROR

```

1573 .SBTTL PTYERR - Signal a parity detect error
1574 ;*****
1575 ;INPUTS: [IE1] The address of the parity CSR associated with the
1576 ; error.
1577 ;
1578 ;OUTPUTS: [AD01] - Parity error messages are displayed on the console.
1579 ;
1580 ;CALLING SEQUENCE: CALL PTYERR,IE1
1581 ;
1582 ;SUBORDINATE ROUTINES CALLED: [CL1]-ERROR [CL2]-WRITE [CL3]-WRIUSN
1583 ; [CL4]-WRITPG
1584 ;*****
1585
1586 004772 000240 PTYERR: NOP ;UFD 03/13/84
1587 004774 PRESERVE ;Preserve registers.
1588 004774 004567 001052 JSR R5,PREG05
1589 005000 004767 002170 JSR PC,TPARER ;Get the address of the error message table.
1590 005004 CALL ERROR,(SP)+ ;Signal the error and wait for Control E.
1591 005004 012605 MOV (SP)+,R5
1592 005006 004767 000432 JSR PC,ERROR
1593
1594 005012 CALL WRITE,0EPACSR ;Display "this is the parity CSR".
1595 005014 012705 002010 MOV 0EPACSR,R5
1596 005016 004767 000562 JSR PC,WRITE
1597
1598 005022 CALL WRIUSN,R5,SLOT(SP),0,08. ;Display the CSR address.
1599 005024 016605 000014 MOV R5,SLOT(SP),R5
1600 005026 012704 000050 MOV 0,R4
1601 005028 012703 000016 MOV 08.,R3
1602 005036 004767 000576 JSR PC,WRIUSN
1603
1604 005042 003767 001052 TST UFDCHN ;ARE WE IN UFD F.S. MODE? ;UFD 03/08/84
1605 005046 001405 BEQ 1$ ;BR IF NOT ;UFD 03/08/84
1606 005050 012737 000032 001104 MOV 032,00UFDCHR ;MAKE LINE #2 TYPED ;UFD 03/08/84
1607 005056 CALL CTRLX ;LET'S GET OUT ;UFD 03/08/84
1608 005056 004767 175204 JSR PC,CTRLX
1609
1610 005062 1$: JSR PC,FGCTU ;UFD 03/08/84
1611 005062 004767 002072 CALL WRITPG,(SP)+ ;Display "continuing with the test"
1612 005066 MOV (SP)+,R5
1613 005070 004767 000440 JSR PC,WRITPG
1614
1615 005074 RETURN
1616 005074 000207 RTS PC

```

SPLADR - SET THE SEGMENT ADDRESS TABLE WITH A POSITIVE ADDRESSI

```

1606 .SBTTL SPLADR- Set the Segment Address Table with a positive addressing pattern
1607 ;*****
1608 ;INPUTS: [IE1] The address bit to be used in developing
1609 ; the address pattern.
1610 ;
1611 ;OUTPUTS: [OO1] - The Segment Address Table is set up with a positive
1612 ; addressing pattern.
1613 ;
1614 ;CALLING SEQUENCE: CALL SPLADR
1615 ;
1616 ;SUBORDINATE ROUTINES CALLED: None
1617 ;*****
1618
1619 005076 000240 SPLADR: NOP ;OPD 03/03/84
1620 005100 PRESERVE ;Preserve registers.
1621 005100 004567 000746 USE R5,PRESERVE
1622 005104 012704 120000 MOV #120000,R3 ;Set the address of the segment address
1623 ;table.
1624 005110 012703 020000 MOV #0,R3 ;Set the address to be used (PAR 1 range).
1625 005114 012702 010000 MOV #4095,R2 ;Set the count.
1626 005120 012701 160000 MOV #16000,R1 ;Establish a mask for a 4K word range.
1627 005124 005000 CLR R0 ;Initialize the address word.
1628
1629 005126 010014 1$: MOV R0,(R4) ;Place the address in the Segment Address
1630 ;table.
1631 005130 060324 ADD R3,(R4) ;
1632
1633 005132 060500 ADD R5,R0 ;Update the address.
1634 005134 030100 BIT R1,R0 ;Check for an overflow.
1635 005136 001403 BEQ 2$ ;
1636
1637 005140 062700 000002 ADD #2,R0 ;If overflow, add in the overflow increment.
1638 005144 040100 BIC R1,R0 ;Mask out the overflow bit.
1639
1640 005146 077211 2$: SCB R2,1$ ;Continue until the table is full.
1641
1642 005150 RETURN
005150 000207 RTS PC

```

SMIADR- SET THE SEGMENT ADDRESS TABLE WITH A NEGATIVE ADDRESSI

```

1644 .SBTTL SMIADR- Set the Segment Address Table with a negative addressing pattern
1645 ;*****
1646 ;INPUTS:      [IE1]  The address bit to be used in developing
1647 ;              the address pattern.
1648 ;
1649 ;OUTPUTS:     [DO1] - The Segment Address Table is set up with a negative
1650 ;              addressing pattern.
1651 ;
1652 ;CALLING SEQUENCE:  CALL  SMIADR,IE1
1653 ;
1654 ;SUBORDINATE ROUTINES CALLED:  None.
1655 ;*****
1656
1657 SMIADR: NOP                ;UFD 03/13/84
1658 PRESERVE                ;Preserve registers.
1659 JSR      R5,PREG05
1660 MOV     #120000,R4       ;Set up the address of the segment address
1661 ;table.
1662 MOV     #20000,R3        ;Set the base address to be used (PAR 1 range).
1663 MOV     #4096.,R2        ;Set word count.
1664 MOV     #160000,R1       ;Establish a mask for a 4K word range.
1665 MOV     #17776,R0        ;Initialize the address word.
1666
1667 1$:  MOV     R0,(R4)      ;Place the address in the Segment Address
1668 ;table.
1669 ADD     R3,(R4)+        ;
1670
1671 SUB     R5,R0            ;Update the address.
1672 BIT     R1,R0            ;Check for an underflow.
1673 BEQ     2$              ;
1674
1675 SUB     #2,R0            ;If underflow, subtract underflow increment.
1676 BIC     R1,R0            ;Mask out the underflow bits.
1677
1678 2$:  SOB     R2,1$       ;Continue until the table is full.
1679
1680 RETURN
      RTS     PC

```

INVERT INVERT A SEGMENT OF MEMORY

1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702 005230 000240
 1703 005232
 005232 004567 000614
 1704
 1705 005236 012705 120000
 1706 005242 005000
 1707
 1708 005244 012703 010000
 1709
 1710 005250 012504
 1711 005252 005114
 1712 005254 061400
 1713 005256 077304
 1714
 1715 005260
 005260 010066 000002
 005264 000207

```
.SBTTL INVERT - Invert a segment of memory
;*****
;INPUTS:      [DI1]  Addresses from the segment address table.
;
;OUTPUTS:     [OE1] - The test flag.  It will be equal to minus 4096 or zero
;               if no errors are detected.
;               [DO1] - The 4K word memory segment determined by the segment
;               address table is inverted.
;
;CALLING SEQUENCE:  CALL  INVERT
;
;COMMENTS:      The memory segment is accessed via the segment address table.
;               Each word in the memory segment should be pointed to by one,
;               and only one element of the segment address table.  Thus the
;               order in which the memory segment is accessed is determined
;               by the ordering of the segment address table.
;
;SUBORDINATE ROUTINES CALLED:  None.
;*****
INVERT:  NOP                ;UFD 03/13/84
        PRESERVE          ;Preserve registers.
        JSR      R5,PREG05
;
        MOV      #120000,R5 ;Starting address of the segment address table.
        CLR      R0        ;Initialize the test flag.
;
        MOV      #4096.,R3 ;Set the word count.
1$:     MOV      (R5)+,R4   ;Get the address to be tested.
        COM      (R4)      ;Invert it.
        ADD      (R4),R0   ;Update the test flag.
        SOB     R3,1$     ;Do the complete segment.
;
        RETURN  R0        ;Return the test flag [OE1].
        MOV     R0,R0$LOT(SP)
        RTS     PC
```

SEGERR - SIGNAL AN ERROR IN A SEGMENT OF MEMORY

```

1717 .SBTTL SEGERR - Signal an error in a segment of memory
1718 ;*****
1719 ;INPUTS:      [DI1]  The contents of PAR 1, the current memory segment
1720 ;              under test.
1721 ;
1722 ;OUTPUTS:     [AD01] - Memory segment error messages are displayed on the
1723 ;              console.
1724 ;
1725 ;CALLING SEQUENCE:  CALL  SEGERR
1726 ;
1727 ;SUBORDINATE ROUTINES CALLED:  [CL1]-ERROR      [CL2]-WRITE      [CL3]-WRIUSN
1728 ;                               [CL4]-WRITPG     [CL5]-SETBAK     [CL6]-MAIN1
1729 ;*****
1730
1731 005266 013700 172342 SEGERR: MOV      @#KPAR1,R0      ;Get the current contents of PAR 1.
1732
1733 005272 004767 001712 JSR      PC,TSEGER      ;Get the address of the message table.
1734 005276 005276 012605 CALL     ERROR,(SP)+    ;Signal the error and wait for Control E.
1735 005300 004767 000140 MOV      (SP)+,R5
1736 005304 012705 002143 JSR      PC,ERROR
1737 005310 004767 000270 CALL     WRITE,#ESEGAD ;Display the "this is the bad segment" message.
1738 005314 010005 000060 MOV      #ESEGAD,R5
1739 005316 012704 000010 JSR      PC,WRITE
1740 005322 012703 000010 CALL     WRIUSN,R0,#'0,#8. ;Display the bad segment address.
1741 005326 004767 000306 MOV      R0,R5
1742 005332 005767 000562 MOV      #'0,R4
1743 005336 001405 000032 MOV      #8,R3
1744 005340 012737 000032 JSR      PC,WRIUSN
1745 005346 004767 174714 TST     UFDCHN          ;ARE WE IN UFD F.S. MODE? ;UFD 03/08/84
1746 005352 004767 001550 BEQ     1$              ;BR IF NOT ;UFD 03/08/84
1747 005356 012605 000150 MOV      #32,@#UFDCHR ;MAKE LIKE #2 TYPED ;UFD 03/08/84
1748 005364 004767 176664 CALL     CTRLX          ;LET'S GET OUT ;UFD 03/08/84
1749 005370 004767 174004 JSR      PC,CTRLX
1750 005370 004767 174004 JSR      PC,TRSTRT      ;Get the address of the "restarting" message ;UFD 03/08/84
1751 005370 004767 174004 ;address table.
1752 005370 004767 174004 CALL     WRITPG,(SP)+   ;Display the message.
1753 005370 012605 000150 MOV      (SP)+,R5
1754 005370 004767 176664 JSR      PC,WRITPG
1755 005370 004767 176664 CALL     SETBAK        ;Ensure the program is ready for restart.
1756 005370 004767 176664 JSR      PC,SETBAK
1757 005370 004767 174004 CALL     MAIN1         ;Go restart the program.
1758 005370 004767 174004 JSR      PC,MAIN1

```

ABORT (ABORT ROUTINE) IF UNDER UFD TRAP UP TO THE MONITOR

```

1753 .SBTTL ABORT - (ABORT ROUTINE) If under UFD trap up to the monitor
1754 ;*****
1755 ;INPUTS: (IE1) SAV30,SAV32 contain the location in the monitor
1756 ; to go to set the dsr error flag
1757 ;
1758 ;CALLING SEQUENCE: JSR PC,ABORT
1759 ;*****
1760
1761 005374 005767 000522 ABORT: TST UFDQUI ;Test for UFD QUIET mode ;UFD 02/08/84
1762 005400 001420 BEQ 10$ ;If not UFD then continue normal operation
1763
1764 005402 004767 176646 CALL SETBAK ;RESTORE PGM TO ORIG LOC. ;UFD 03/09/84
1765 005402 JSR PC,SETBAK
1766 005406 016737 000512 000030 MOV SAV30,@030 ;Restore EMT location 30
1767 005414 016737 000506 000032 MOV SAV32,@032 ;Restore EMT priority location 32
1768 005422 104042 EMT +42 ;Get DCA into R0 from monitor
1769 005424 012760 177777 000042 MOV @-1,42(R0) ;Set @ -1 into location dserr in monitor
1770
1771 005432 005067 172404 CLR 42 ;Clear monitor return flag
1772 005436 004777 000452 JSR PC,@XXRTN ;Return to XXDP.
1773 005442 000207 10$: RTS PC ;If not UFD return to mainline
1774

```

ERROR - HANDLE STANDARD ERROR ACTIONS

```

1776      ,SBTTL ERROR   Handle standard error actions
1777      ;*****
1778      ;INPUTS:      [IE1]   The address of the table of messages to be displayed
1779      ;                if the operator inputs a Control E.
1780      ;
1781      ;OUTPUTS:     [AD01] - The standard customer error message is displayed on
1782      ;                the console.
1783      ;                [+AD02]- Expanded error information based on IE1 is displayed
1784      ;                on the console
1785      ;
1786      ;CALLING SEQUENCE:  CALL   ERROR,IE1
1787      ;
1788      ;SUBORDINATE ROUTINES CALLED:  [CL1]-WRITPG   [CL2]-GETCHR   [+CL3]-WRITPG
1789      ;                                [+CL4]-CTRLX
1790      ;*****
1791
1792 005444 000240      ERROR:  NOP                                ;UFD 03/13/84
1793 005446      PRESERVE
1794 005446 004567 000400      JSR      R5,PREG05      ;Preserve registers.
1795 005452 004767 177716      JSR      PC,ABORT      ;Check if we are under UFD ?
1796 005456 004767 001416      JSR      PC,TCUSTM     ;Get the address of the table containing the
1797 005462      CALL     WRITPG,(SP),      ;customer messages and display it.
1798 005462 012605      MOV      (SP),R5
1799 005464 004767 000044      JSR      PC,WRITPG
1800 005470      1$:    CALL     GETCHR      ;Get an input character.
1801 005474 020027 000030      JSR      PC,GETCHR
1802 005500 001002      CMP      R0,#30      ;Check for Control X.
1803 005502      BNE     2$
1804 005502 004767 174560      CALL     CTRLX      ;If Control X, exit via CTRLX.
1805 005502      JSR      PC,CTRLX
1806 005506 005767 000406      2$:    TST      UFDCHN      ;ARE WE IN UFD F.S. MODE?      ;UFD 03/08/84
1807 005512 001003      BNE     3$      ;BR IF WE ARE      ;UFD 03/08/84
1808 005514 020027 000005      CMP      R0,#5      ;Check for Control E.
1809 005520 001363      BNE     1$      ;If not Control E, wait in loop.
1810 005522      3$:    CALL     WRITPG,R5SLOT(SP) ;Display the expanded error information.
1811 005522 016605 000014      MOV      R5SLOT(SP),R5
1812 005526 004767 000002      JSR      PC,WRITPG
1813 005532      RETURN
1814 005532 000207      RTS      PC

```

WRITPG (WRITE PACKAGE) WRITE SEVERAL ASCII2 STRINGS WITH RETU

```

1813 .SHTTL WRITPG (WRITE PACKAGE) Write several ASCII2 strings with returns
1814 ;*****
1815 ;INPUTS: [IE1] R5 contains a table address. That table contains
1816 ; the offsets (into the text table) of the strings to be
1817 ; printed in order. The table must be terminated by a zero.
1818 ;
1819 ;OUTPUTS: [AD01] The strings are displayed on the console.
1820 ;
1821 ;CALLING SEQUENCE: CALL WRITPG,IE1
1822 ;
1823 ;SUBORDINATE ROUTINES CALLED: WRITLN.
1824 ;*****
1825
1826 005534 WRITPG: PUSH R5,R4 ;Save registers.
      005534 010546 MOV R5,-(SP)
      005536 010446 MOV R4,-(SP)
1827
1828 005540 010504 MOV R5,R4 ;Put the table address in R4.
1829
1830 005542 012405 1$: MOV (R4)+,R5 ;Put the string offset in R5 for WRITLN.
1831 005544 001403 PEQ 2$ ;If terminator found, get out.
1832
1833 005546 CALL WRITLN ;Print the string as a line.
      005546 004767 000010 JSR PC,WRITLN
1834 005552 000773 BR 1$ ;And get the next string.
1835
1836 005554 2$: POP R4,R5 ;Restore registers.
      005554 012604 MOV (SP)+,R4
      005556 012605 MOV (SP)+,R5
1837 005560 RETURN
      005560 000207 RTS PC

```

WRITE (WRITE PACKAGE) WRITE AN ASCIZ STRING TO THE CONSOLE

```

1861 .SBTTL WRITE - (WRITE PACKAGE) Write an ASCIZ string to the console
1862 ;*****
1863 ;INPUTS: [IE1] R5 contains the offset (into the text area) of the
1864 ; ASCIZ string.
1865 ;
1866 ;OUTPUTS: [AD01] - The ASCIZ string is written to the console.
1867 ;
1868 ;CALLING SEQUENCE CALL WRITE,IE1
1869 ;
1870 ;SUBORDINATE ROUTINES CALLED: [CL1]-PRTCHR.
1871 ;*****
1872
1873 005604 WRITE: PUSH R5,R4 ;Save registers.
      005604 010546 MOV R5,-(SP)
      005606 010446 MOV R4,-(SP)
1874
1875 005610 004767 001512 JSR PC,LOCIXT ;Get the text area address.
1876 005614 012604 MOV (SP)+,R4 ;Put it in R4.
1877 005616 060504 ADD R5,R4 ;Add in the offset.
1878
1879 005620 112405 1$: MOV (R4)+,R5 ;Put a byte in R5 for PRTCHR.
1880 005622 001403 BEQ 2$ ;If end of string, get out.
1881
1882 005624 CALL PRTCHR ;If not, print the character.
      005624 004767 174310 JSR PC,PRTCHR
1883 005630 000773 BR 1$ ;And get the next byte.
1884
1885 005632 2$: POP R4,R5 ;Restore registers.
      005632 012604 MOV (SP)-,R4
      005634 012605 MOV (SP)+,R5
1886 005636 RETURN
      005636 000207 RTS PC

```

WRIUSN - (WRITE PACKAGE.) WRITE AN UNSIGNED NUMBER TO THE CONSO

```

1888 .SBTTL WRIUSN - (WRITE PACKAGE) Write an unsigned number to the console
1889 ;*****
1890 ;INPUTS:      [IE1]  R5 contains the number in binary.
1891 ;              [IE2]  R4 contains 60 if leading zeros are to be printed,
1892 ;              40 if leading spaces are to be printed, zero
1893 ;              if neither.
1894 ;              WARNING - If IE1 and IE2 are both zero, nothing will
1895 ;              be printed.
1896 ;              [IE3]  R3 contains 10 (base 10) if the number is to be
1897 ;              converted to Decimal, 8 if octal. Default is binary.
1898 ;
1899 ;OUTPUTS:     [AD01] The number is displayed on the console terminal.
1900 ;
1901 ;CALLING SEQUENCE:  CALL  WRIUSN,IE1,IE2,IE3
1902 ;
1903 ;COMMENTS:     This is a two page module.
1904 ;
1905 ;SUBORDINATE ROUTINES CALLED:  [CL1]-PRTCHR.
1906 ;*****
1907 005640 000240 WRIUSN: NOP ;UFD 03/13/84
1908 005642 PRESERVE ;Preserve registers.
1909 005642 004567 000204 JSR R5,PREG05
1910 005646 010500 MOV R5,R0 ;Put the number in R0.
1911 005650 005046 CLR -(SP) ;Create a slot for the character count.
1912 005652 004737 000044 JSR PC,3$ ;Set up for the local co-routine call.
1913 005656 022703 000010 CMP #8.,R3 ;Check for octal printing.
1914 005662 001434 BEQ 5$ ;Go handle octal.
1915 005664 022703 000012 CMP #10.,R3 ;Check for decimal printing.
1916 005670 001434 BEQ 6$ ;Go handle decimal.
1917
1918 005672 012701 000020 MOV #16.,R1 ;Set loop count to print 16 bits in binary.
1919 005676 005075 1$: CLR R5 ;
1920 005700 006100 ROL R0 ;Put a bit in R5.
1921 005702 006105 ROL R5 ;
1922 005704 004736 JSR PC,@(SP)+ ;The co-routine call prints the bit.
1923 005706 077105 SOB R1,1$ ;Print 16 bits.
1924
1925 005710 2$: POP R0 ;Purge the stack of the co-routine call.
1926 005710 012600 MOV (SP)+,R0
1927 005712 POP R0 ;Get the character count in R0.
1928 005712 012600 MOV (SP)+,R0
1929 005714 RETURN R0 ;Return the character count.
1930 005714 010066 000002 MOV R0,R0SLOT(SP)
1931 005720 000207 RTS PC
1932
1933 ;***
1934 ;This co-routine converts to ASCII and prints
1935 ;***
1936 005722 004736 3$: JSR PC,@(SP)+ ;Return from the co-routine.
1937
1938 005724 005705 TST R5 ;Check the number for zero.
1939 005726 001402 BEQ 4$ ;
1940 005730 012704 000060 MOV #60,R4 ;If not zero, set to print any later zeros.
1941
1942 005734 005704 4$: TST R4 ;Check the print flag.

```

WRIUSN - (WRITE PACKAGE) WRITE AN UNSIGNED NUMBER TO THE CONSO

1940	005736	001771		DEQ	3\$;If zero, don't print.
1941	005740	060405		ADD	R4,R5		;Convert to ASCII.
1942	005742			CALL	PRTCHR		;Print the character.
	005742	004767	174172	JSR	PC,PRTCHR		
1943	005746	005266	000002	INC	2(SP)		;Bump the character count.
1944	005752	000763		BR	3\$;Return from the co-routine.

WRIUSN - (WRITE PACKAGE) WRITE AN UNSIGNED NUMBER TO THE CONSO

```

1946
1947
1948
1949
1950
1951
1952 005754 004767 000034 5$: JSR PC,11$ ;Get the address of the octal conversion table.
1953 005760 000402 BR 7$ ;
1954
1955 005762 004767 000046 6$: JSR PC,12$ ;Get the address of the decimal table.
1956
1957 005766 012602 7$: MOV (SP)+,R2 ;Put the table address in R2.
1958 005770 000401 BR 9$ ;Go do the conversion.
1959
1960 005772 004736 8$: JSR PC,@(SP)+ ;Convert to ASCII and print.
1961
1962 005774 012201 9$: MOV (R2)+,R1 ;Get the next base from the table.
1963 005776 001744 BEQ 2$ ;If finished, get out.
1964
1965 006000 005005 10$: CLR R5 ;Initialize the digit to be printed.
1966 006002 020100 CMP R1,R0 ;Compare the base with the remaining number.
1967 006004 101372 BHI 8$ ;If the base is higher, go print the number.
1968
1969 006006 005205 INC R5 ;If not, bump the number.
1970 006010 160100 SUB R1,R0 ;Subtract the base from the number.
1971 006012 000773 BR 10$ ;Loop until the base is higher.
1972
1973
1974 006014 004736 11$: JSR PC,@(SP)+ ;Return the address of the octal table.
1975 006016 100000 .WORD 100000 ;
1976 006020 010000 .WORD 10000 ;
1977 006022 001000 .WORD 1000 ;
1978 006024 000100 .WORD 100 ;
1979 006026 000010 .WORD 10 ;
1980 006030 000001 .WORD 1 ;
1981 006032 000000 .WORD 0 ;Table terminator.
1982
1983 006034 004736 12$: JSR PC,@(SP)+ ;Return the address of the decimal table.
1984 006036 023420 .WORD 10000, ;
1985 006040 001750 .WORD 1000, ;
1986 006042 000144 .WORD 100, ;
1987 006044 000012 .WORD 10, ;
1988 006046 000001 .WORD 1 ;
1989 006050 000000 .WORD 0 ;Table terminator.

```

PREG05 - PRESERVE REGISTERS R0 THROUGH R5 FOR SUBROUTINE CALLS

```

1991      ,SBTTL PREG05 - Preserve Registers R0 through R5 for subroutine calls
1992      ;*****
1993      ;INPUTS:      The return address from the caller's caller must be the last
1994      ;                entry on the stack.
1995      ;
1996      ;OUTPUTS:     Registers R0 through R5 are saved on the stack. When
1997      ;                the return is made from the calling subroutine, they are
1998      ;                automatically restored.
1999      ;
2000      ;CALLING SEQUENCE:  PRESERVE
2001      ;
2002      ;COMMENTS:     This routine is re-entrant.
2003      ;
2004      ;                Parameters may be passed out of a subroutine by using the
2005      ;                "RETURN" macro.
2006      ;
2007      ;SUBORDINATE ROUTINES CALLED:  None.
2008      ;*****
2009
2010      006052      PREG05:                ;R5 has been loaded on the stack by the call.
2011      ;R5 now contains the return to the calling routine.
2012
2013      006052      PUSH      R4,R3,R2,R1,R0,R5
2014      006052      010446      MOV      R4,-(SP)
2015      006054      010346      MOV      R3,-(SP)
2016      006056      010246      MOV      R2,-(SP)
2017      006060      010146      MOV      R1,-(SP)
2018      006062      010046      MOV      R0,-(SP)
2019      006064      010546      MOV      R5,-(SP)
2020      006066      016605      MOV      R5,SLOT(SP),R5 ;Restore R5 to its original value.
2021
2022      006072      004736      JSR      PC,@(SP)+ ;Call the calling routine.
2023
2024      ;+++
2025      ;The following code is executed when the calling routine
2026      ;does a "RETURN".
2027      ;---
2028
2029      006074      POP      R0,R1,R2,R3,R4,R5
2030      006074      012600      MOV      (SP)+,R0
2031      006076      012601      MOV      (SP)+,R1
2032      006100      012602      MOV      (SP)+,R2
2033      006102      012603      MOV      (SP)+,R3
2034      006104      012604      MOV      (SP)+,R4
2035      006106      012605      MOV      (SP)+,R5
2036
2037      006110      RETURN      ;Return to the caller's caller.
2038      006110      000207      RTS      PC

```

GLOBALS

```

2027      .SBTTL  GLOBALS
2028
2029
2030 006112 000000      SWADD:  .WORD  0          ;Will contain the address of SWSTAC in XXDP+.
2031
2032 006114 000000      XXRTN:  .WORD  0          ;Will contain the XXDP+ return address/Chain
2033                                     ;Mode flag.
2034
2035 006116 000000      RELFLG: .WORD  0          ;Relocation flag.
2036
2037 006120 000000      UFDCHN: .WORD  0          ;UFD CHAIN MODE flag
2038 006122 000000      UFDQUI: .WORD  0          ;UFD QUIET MODE FLAG          ;UFD 02/08/84
2039
2040 006124 000000      SAV30:  .WORD  0          ;Will contain contents of EMT vector
2041 006126 000000      SAV32:  .WORD  0
2042 006130 000000      SAV52:  .WORD  0          ;SAVE FOR LOC 52          ;UFD 02/08/84
2043 006132 000000      SAVSP:  .WORD  0          ;SAVE FOR SP            ;UFD 02/08/84
2044 006134 000000      SAVCAC: .WORD  0          ;SAVE FOR CACHE         ;UFD 04/03/84
2045 006136 000000      ORIONF: .WORD  0          ;ORION CP FLAG          ;UFD 04/03/84
2046
2047
2048                                     ;+++
2049                                     ;These are used by the Write Package utility modules.
2050                                     ;---
2051 006140 177560      KBCSR:  .WORD  DEFCON      ;Keyboard CSR.
2052 006142 177562      KBBUF:  .WORD  DEFCON+2    ;Keyboard buffer.
2053 006144 177564      DSPCSR: .WORD  DEFCON+4    ;Display CSR.
2054 006146 177566      DSPBUF: .WORD  DEFCON+6    ;Display buffer.
2055

```

TKMAP1 - (DATA SECTION) - INITIAL MEMORY MANAGEMENT AND PSW MA

```

2057          .SBTTL TKMAP1 - (Data section) - Initial memory management and PSW map
2058
2059 006150 004736 TKMAP1: JSR PC,@(SP)+ ;Table locator.
2060
2061 006152 004406          .WORD RW256 ;256 words, Read/Write.
2062 006154 172300          .WORD KPDR0 ;Address of PDR 0.
2063 006156 077406          .WORD RW4096 ;4096 words, Read/Write.
2064 006160 172302          .WORD KPDR1 ;Address of PDR 1.
2065 006162 077406          .WORD RW4096 ;4096 words, Read/Write.
2066 006164 172304          .WORD KPDR2 ;Address of PDR 2.
2067 006166 077406          .WORD RW4096 ;4096 words, Read/Write.
2068 006170 172306          .WORD KPDR3 ;Address of PDR 3.
2069 006172 077406          .WORD RW4096 ;4096 words, Read/Write.
2070 006174 172310          .WORD KPDR4 ;Address of PDR 4.
2071 006176 077406          .WORD RW4096 ;4096 words, Read/Write.
2072 006200 172312          .WORD KPDR5 ;Address of PDR 5.
2073 006202 077406          .WORD RW4096 ;4096 words, Read/Write.
2074 006204 172314          .WORD KPDR6 ;Address of PDR 6.
2075 006206 077406          .WORD RW4096 ;4096 words, Read/Write.
2076 006210 172316          .WORD KPDR7 ;Address of PDR 7.
2077
2078 005212 001000          .WORD PAR16K ;Base address at 16K.
2079 006214 172340          .WORD KPAR0 ;Address of PAR 0.
2080 006216 000000          .WORD PAR0K ;Base address at 0.
2081 006220 172342          .WORD KPAR1 ;Address of PAR 1.
2082 006222 000200          .WORD PAR4K ;Base address at 4K.
2083 006224 172344          .WORD KPAR2 ;Address of PAR 2.
2084 006226 000400          .WORD PAR8K ;Base address at 8K.
2085 006230 172346          .WORD KPAR3 ;Address of PAR 3.
2086 006232 001000          .WORD PAR16K ;Base address at 16K.
2087 006234 172350          .WORD KPAR4 ;Address of PAR 4.
2088 006236 001200          .WORD PAR20K ;Base address at 20K.
2089 006240 172352          .WORD KPAR5 ;Address of PAR 5.
2090 006242 001400          .WORD PAR24K ;Base address at 24K.
2091 006244 172354          .WORD KPAR6 ;Address of PAR 6.
2092 006246 177600          .WORD PARI0 ;Base address at I/O page.
2093 006250 172356          .WORD KPAR7 ;Address of PAR 7.
2094
2095 006252 000020          .WORD MMU22A ;Enable 22 bit mapping.
2096 006254 172516          .WORD MMUSR3 ;Address of MMU SR3.
2097 006256 000001          .WORD MMUENA ;Enable memory management.
2098 006260 177572          .WORD MMUSRO ;Address of MMU SRO.
2099
2100 006262 000000          .WORD 0 ;Processor priority zero, Kernal mode.
2101 006264 177776          .WORD PSW ;Address of PSW.
2102
2103 006266 000000          .WORD 0 ;
2104 006270 000000          .WORD 0 ;Table terminator.

```

TKMAP2 - (DATA SECTION) - MEMORY MANAGEMENT MAP FOR LOW 16K OP

.SBTTL TKMAP2 - (Data section) - Memory management map for low 16K operation

```

2106
2107
2108
2109
2110 006272 004736      TKMAP2: JSR      PC,@(SP)+      ;Table locator.
2111
2112 006274 000000      .WORD      PAR0K      ;Base address at 0.
2113 006276 172340      .WORD      KPAR0      ;Address of PAR 0.
2114 006300 001000      .WORD      PAR16K     ;Base address at 16K.
2115 006302 172342      .WORD      KPAR1      ;Address of PAR 1.
2116 006304 001200      .WORD      PAR20K     ;Base address at 20K.
2117 006306 172344      .WORD      KPAR2      ;Address of PAR 2.
2118 006310 001400      .WORD      PAR24K     ;Base address at 24K.
2119 006312 172346      .WORD      KPAR3      ;Address of PAR 3.
2120 006314 000000      .WORD      PAR0K      ;Base address at 0.
2121 006316 172350      .WORD      KPAR4      ;Address of PAR 4.
2122 006320 000200      .WORD      PAR4K      ;Base address at 4K.
2123 006322 172352      .WORD      KPAR5      ;Address of PAR 5.
2124 006324 000400      .WORD      PAR8K      ;Base address at 8K.
2125 006326 172354      .WORD      KPAR6      ;Address of PAR 6.
2126 006330 000000      .WORD      0          ;
2127 006332 000000      .WORD      0          ;Table terminator.

```

TKMAP3 - (DATA SECTION) MEMORY MANAGEMENT MAP FOR TESTING

```

2129          .SBTTL  TKMAP3 - (Data section) Memory Management Map for testing
2130
2131
2132 006334 004736      TKMAP3: JSR      PC,@(SP)+      ;Table locator.
2133
2134 006336 077400          .WORD      NR4096          ;Non-Resident.
2135 006340 172304          .WORD      KPDR2            ;Address of PDR 2.
2136 006342 077400          .WORD      NR4096          ;Non-Resident.
2137 006344 172306          .WORD      KPDR3            ;Address of PDR 3.
2138 006346 077402          .WORD      R04096          ;Read only.
2139 006350 172310          .WORD      KPDR4            ;Address of PDR 4.
2140 006352 077402          .WORD      R04096          ;Read only.
2141 006354 172312          .WORD      KPDR5            ;Address of PDR 5.
2142 006356 077400          .WORD      NR4096          ;Non-Resident.
2143 006360 172314          .WORD      KPDR6            ;Address of PDR 6.
2144 006362 000000          .WORD      0                ;
2145 006364 000000          .WORD      0                ;Table terminator.

```

TKMAP4 - (DATA SECTION) MEMORY MANAGEMENT MAP TO RESET AFTER T

```

2147          .SBTTL  TKMAP4 - (Data section) Memory Management Map to reset after testing
2148
2149
2150 006366  000000          TKMAP4: JSR      PC,@(SP)+      ;Table locator.
2151
2152 006370  077406          .WORD      RW4096          ;4096 words, Read/Write.
2153 006372  172304          .WORD      KPDR2           ;Address of PDR 2.
2154 006374  077406          .WORD      RW4096          ;4096 words, Read/Write.
2155 006376  172306          .WORD      KPDR3           ;Address of PDR 3.
2156 006400  077406          .WORD      RW4096          ;4096 words, Read/Write.
2157 006402  172310          .WORD      KPDR4           ;Address of PDR 4.
2158 006404  077406          .WORD      RW4096          ;4096 words, Read/Write.
2159 006406  172312          .WORD      KPDR5           ;Address of PDR 5.
2160 006410  077406          .WORD      RW4096          ;4096 words, Read/Write.
2161 006412  172314          .WORD      KPDR6           ;Address of PDR 6.
2162 006414  000000          .WORD      0              ;
2163 006416  000000          .WORD      0              ;Table terminator.

```

VCTORS (DATA AREA) VECTORS TABLE FOR UNXINT

Address	Offset	PC	Instruction	Target	Comment
2165					.SBTTL VCTORS - (Data area) Vectors table for UNXINT
2166					
2167					
2168					
2169	006420	004736	VCTORS:	JSR PC,8(SP)+	;Table locator.
2170					
2171	006422	004767	173312	VCT000:	JSR PC,UNXINT
2172	006426	004767	173306	VCT004:	JSR PC,UNXINT
2173	006432	004767	173302	VCT010:	JSR PC,UNXINT
2174	006436	004767	173276	VCT014:	JSR PC,UNXINT
2175	006442	004767	173272	VCT020:	JSR PC,UNXINT
2176	006446	004767	173266	VCT024:	JSR PC,UNXINT
2177	006452	004767	173262	VCT030:	JSR PC,UNXINT
2178	006456	004767	173256	VCT034:	JSR PC,UNXINT
2179					
2180	006462	004767	173252	VCT040:	JSR PC,UNXINT
2181	006466	004767	173246	VCT044:	JSR PC,UNXINT
2182	006472	004767	173242	VCT050:	JSR PC,UNXINT
2183	006476	004767	173236	VCT054:	JSR PC,UNXINT
2184	006502	004767	173232	VCT060:	JSR PC,UNXINT
2185	006506	004767	173226	VCT064:	JSR PC,UNXINT
2186	006512	004767	173222	VCT070:	JSR PC,UNXINT
2187	006516	004767	173216	VCT074:	JSR PC,UNXINT
2188					
2189	006522	004767	173212	VCT100:	JSR PC,UNXINT
2190	006526	004767	173206	VCT104:	JSR PC,UNXINT
2191	006532	004767	173202	VCT110:	JSR PC,UNXINT
2192	006536	004767	173176	VCT114:	JSR PC,UNXINT
2193	006542	004767	173172	VCT120:	JSR PC,UNXINT
2194	006546	004767	173166	VCT124:	JSR PC,UNXINT
2195	006552	004767	173162	VCT130:	JSR PC,UNXINT
2196	006556	004767	173156	VCT134:	JSR PC,UNXINT
2197					
2198	006562	004767	173152	VCT140:	JSR PC,UNXINT
2199	006566	004767	173146	VCT144:	JSR PC,UNXINT
2200	006572	004767	173142	VCT150:	JSR PC,UNXINT
2201	006576	004767	173136	VCT154:	JSR PC,UNXINT
2202	006602	004767	173132	VCT160:	JSR PC,UNXINT
2203	006606	004767	173126	VCT164:	JSR PC,UNXINT
2204	006612	004767	173122	VCT170:	JSR PC,UNXINT
2205	006616	004767	173116	VCT174:	JSR PC,UNXINT

D6

VECTORS - (DATA AREA) VECTORS TABLE FOR UNXINT

2207					
2208	006622	004767	173112	VCT200: JSR	PC, UNXINT
2209	006626	004767	173106	VCT204: JSR	PC, UNXINT
2210	006632	004767	173102	VCT210: JSR	PC, UNXINT
2211	006636	004767	173076	VCT214: JSR	PC, UNXINT
2212	006642	004767	173072	VCT220: JSR	PC, UNXINT
2213	006646	004767	173066	VCT224: JSR	PC, UNXINT
2214	006652	004767	173062	VCT230: JSR	PC, UNXINT
2215	006656	004767	173056	VCT234: JSR	PC, UNXINT
2216					
2217	006662	004767	173052	VCT240: JSR	PC, UNXINT
2218	006666	004767	173046	VCT244: JSR	PC, UNXINT
2219	006672	004767	173042	VCT250: JSR	PC, UNXINT
2220	006676	004767	173036	VCT254: JSR	PC, UNXINT
2221	006702	004767	173032	VCT260: JSR	PC, UNXINT
2222	006706	004767	173026	VCT264: JSR	PC, UNXINT
2223	006712	004767	173022	VCT270: JSR	PC, UNXINT
2224	006716	004767	173016	VCT274: JSR	PC, UNXINT
2225					
2226	006722	004767	173012	VCT300: JSR	PC, UNXINT
2227	006726	004767	173006	VCT304: JSR	PC, UNXINT
2228	006732	004767	173002	VCT310: JSR	PC, UNXINT
2229	006736	004767	172776	VCT314: JSR	PC, UNXINT
2230	006742	004767	172772	VCT320: JSR	PC, UNXINT
2231	006746	004767	172766	VCT324: JSR	PC, UNXINT
2232	006752	004767	172762	VCT330: JSR	PC, UNXINT
2233	006756	004767	172756	VCT334: JSR	PC, UNXINT
2234					
2235	006762	004767	172752	VCT340: JSR	PC, UNXINT
2236	006766	004767	172746	VCT344: JSR	PC, UNXINT
2237	006772	004767	172742	VCT350: JSR	PC, UNXINT
2238	006776	004767	172736	VCT354: JSR	PC, UNXINT
2239	007002	004767	172732	VCT360: JSR	PC, UNXINT
2240	007006	004767	172726	VCT364: JSR	PC, UNXINT
2241	007012	004767	172722	VCT370: JSR	PC, UNXINT
2242	007016	004767	172716	VCT374: JSR	PC, UNXINT

MESSAGE TABLES (DATA SECTION)

```

2244      .SBTTL MESSAGE TABLES (Data section)
2245
2246 007022 004736      TKTERR: JSR      PC,0(SP)+      ;Table locator.
2247
2248 007024 000103      .WORD      MEXINF      ;Expanded error message lines.
2249 007026 000177      .WORD      EKTER1      ;
2250 007030 000310      .WORD      EKTER2      ;Signal a Memory management abort.
2251 007032 000422      .WORD      EKTER3      ;
2252 007034 000511      .WORD      MCTUT1      ;
2253 007036 000547      .WORD      MCTUT2      ;
2254 007040 000001      .WORD      NEWLIN      ;
2255 007042 000000      .WORD      0           ;
2256
2257 007044 004736      TLOW32: JSR      PC,0(SP)+      ;Table locator.
2258
2259 007046 000103      .WORD      MEXINF      ;Expanded error message lines.
2260 007050 000614      .WORD      EL32M1      ;
2261 007052 000725      .WORD      EL32M2      ;Signal an error in the low 32K words of
2262 007054 000511      .WORD      MCTUT1      ;memory.
2263 007056 000547      .WORD      MCTUT2      ;
2264 007060 000001      .WORD      NEWLIN      ;
2265 007062 000000      .WORD      0           ;
2266
2267 007064 004736      TNOPTY: JSR      PC,0(SP)+      ;Table locator.
2268
2269 007066 000103      .WORD      MEXINF      ;Expanded error message lines.
2270 007070 001026      .WORD      ENOPT1      ;
2271 007072 000511      .WORD      MCTUT1      ;Signal a no parity detect situation.
2272 007074 000001      .WORD      NEWLIN      ;
2273 007076 000000      .WORD      0           ;
2274
2275 007100 004736      TCUSTM: JSR      PC,0(SP)+      ;Table locator.
2276
2277 007102 000001      .WORD      NEWLIN      ;
2278 007104 000001      .WORD      NEWLIN      ;
2279 007106 000001      .WORD      NEWLIN      ;Standard customer error message.
2280 007110 001117      .WORD      ECUST1      ;
2281 007112 000001      .WORD      NEWLIN      ;
2282 007114 001221      .WORD      ECUST2      ;
2283 007116 000001      .WORD      NEWLIN      ;
2284 007120 001261      .WORD      ECUST3      ;
2285 007122 001367      .WORD      ECUST4      ;
2286 007124 000000      .WORD      0           ;

```

SYMBOL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES
UFDCHR	001104	*10-385 *11-445 *15-638 19-749 *19-756 *19-761 19-762 19-768 *38-1565
UFDQUT	006122	*39-1598 *43-1742
UNXERR	004666	*11-434 15-623 17-685 44-1761 *52-2038
UNXINT	001740	14-598 *38-1553
		*14-590 57-2171 57-2172 57-2173 57-2174 57-2175 57-2176 57-2177 57-2178
		57-2180 57-2181 57-2182 57-2183 57-2184 57-2185 57-2186 57-2187 57-2189
		57-2190 57-2191 57-2192 57-2193 57-2194 57-2195 57-2196 57-2198 57-2199
		57-2200 57-2201 57-2202 57-2203 57-2204 57-2205 58-2208 58-2209 58-2210
		58-2211 58-2212 58-2213 58-2214 58-2215 58-2217 58-2218 58-2219 58-2220
		58-2221 58-2222 58-2223 58-2224 58-2226 58-2227 58-2228 58-2229 58-2230
		58-2231 58-2232 58-2233 58-2235 58-2236 58-2237 58-2238 58-2239 58-2240
		58-2241 58-2242
VCTORS	006420	14-590 20-819 *57-2169
VCT000	006422	*57-2171
VCT004	006426	*57-2172
VCT010	006432	*57-2173
VCT014	006436	*57-2174
VCT020	006442	*57-2175
VCT024	006446	*57-2176
VCT030	006452	*57-2177
VCT034	006456	*57-2178
VCT040	006462	*57-2180
VCT044	006466	*57-2181
VCT050	006472	*57-2182
VCT054	006476	*57-2183
VCT060	006502	*57-2184
VCT064	006506	*57-2185
VCT070	006512	*57-2186
VCT074	006516	*57-2187
VCT100	006522	*57-2189
VCT104	006526	*57-2190
VCT110	006532	*57-2191
VCT114	006536	13-566 *57-2192
VCT120	006542	*57-2193
VCT124	006546	*57-2194
VCT130	006552	*57-2195
VCT134	006556	*57-2196
VCT140	006562	*57-2198
VCT144	006566	*57-2199
VCT150	006572	*57-2200
VCT154	006576	*57-2201
VCT160	006602	*57-2202
VCT164	006606	*57-2203
VCT170	006612	*57-2204
VCT174	006616	*57-2205
VCT200	006622	*58-2208
VCT204	006626	*58-2209
VCT210	006632	*58-2210
VCT214	006636	*58-2211
VCT220	006642	*58-2212
VCT224	006646	*58-2213
VCT230	006652	*58-2214

SYMBOL CROSS REFERENCE

REF V01

SYMBOL	VALUE	REFERENCES
VCT234	006656	058-2215
VCT240	006662	058-2217
VCT244	006666	058-2218
VCT250	006672	058-2219
VCT254	006676	058-2220
VCT260	006702	058-2221
VCT264	006706	058-2222
VCT270	006712	058-2223
VCT274	006716	058-2224
VCT300	006722	058-2226
VCT304	006726	058-2227
VCT310	006732	058-2228
VCT314	006736	058-2229
VCT320	006742	058-2230
VCT324	006746	058-2231
VCT330	006752	058-2232
VCT334	006756	058-2233
VCT340	006762	058-2235
VCT344	006766	058-2236
VCT350	006772	058-2237
VCT354	006776	058-2238
VCT360	007002	058-2239
VCT364	007006	058-2240
VCT370	007012	058-2241
VCT374	007016	058-2242
WRITE	005604	11-481 11-485 11-511 22-910 22-923 22-931 38-1559 39-1592 43-1736 47-1854 47-1856 048-1873
WRITLN	005562	46-1833 047-1852
WRITPG	005534	38-1569 39-1602 43-1747 45-1797 45-1809 046-1826
WRIUSN	005640	22-908 22-918 22-926 38-1561 39-1594 43-1738 049-1907
WRNGPA	007220	23-964 061-2331
XOFF	= 000023	05-232 17-696
XON	= 000021	05-231 17-701
XXRTN	006114	*11-418 11-494 11-503 *20-815 44-1772 052-2032

C/

MACRO CROSS REFERENCE

CREF VOL

MACRO NAME	REFERENCES										
CALL	07-315	11-444	11-450	11-481	11-483	11-485	11-511	12-530	14-598	15-639	
	17-695	17-700	18-723	18-725	18-727	18-729	18-731	18-733	19-753	19-760	
	19-762	19-763	20-814	20-817	20-837	21-881	22-908	22-910	22-919	22-923	
	22-926	22-931	22-937	22-942	23-966	23-974	23-999	24-1018	25-1054	25-1055	
	26-1078	26-1080	26-1082	26-1084	27-1125	28-1183	29-1214	30-1242	30-1246	31-1276	
	31-1280	31-1282	31-1286	32-1311	32-1314	33-1343	33-1349	36-1467	36-1472	36-1480	
	36-1483	36-1487	36-1493	37-1536	38-1557	38-1559	38-1561	38-1566	38-1569	39-1590	
	39-1592	39-1594	39-1599	39-1602	43-1734	43-1736	43-1738	43-1743	43-1747	43-1749	
	43-1751	44-1764	45-1797	45-1799	45-1802	45-1809	46-1833	47-1854	47-1856	48-1882	
	49-1942										
	POP	07-289	12-534	13-565	13-574	14-600	17-705	23-998	28-1178	46-1836	47-1858
		48-1885	49-1925	49-1926	51-2023						
	PRESER	06-254	21-858	22-904	23-960	25-1049	26-1073	28-1153	29-1205	30-1233	31-1268
		32-1307	33-1337	34-1391	35-1422	36-1461	37-1509	38-1554	39-1587	40-1620	41-1658
42-1703		45-1793	49-1908								
PUSH RETURN	06-263	12-527	13-556	17-689	23-968	28-1161	46-1826	47-1852	48-1873	51-2013	
	08-344	16-669	17-687	17-706	18-735	20-843	21-883	22-943	23-1000	25-1057	
	26-1086	27-1136	28-1185	29-1217	30-1252	31-1290	32-1318	33-1363	34-1401	35-1438	
	36-1494	37-1538	38-1571	39-1604	40-1642	41-1680	42-1715	45-1811	46-1837	47-1859	
	48-1886	49-1927	51-2025								

6

CVMSBBO 0-2M QUICK V.....B1
.....C1
.....D1
.....E1
.....F1
.....G1
.....H1
.....I1
.....J1
.....K1
.....L1
.....M1
.....N1

.....B2
.....C2
.....D2
.....E2
.....F2
.....G2
.....H2
.....I2
.....J2
.....K2
.....L2
.....M2
.....N2

.....B3
.....C3
.....D3
.....E3
.....F3
.....G3
.....H3
.....I3
.....J3
.....K3
.....L3
.....M3
.....N3

.....B4
.....C4
.....D4
.....E4
.....F4
.....G4
.....H4
.....I4
.....J4
.....K4
.....L4
.....M4
.....N4

.....B5
.....C5
.....D5
.....E5
.....F5
.....G5
.....H5
.....I5
.....J5
.....K5
.....L5
.....M5
.....N5

.....B6
.....C6
.....D6
.....E6
.....F6
.....G6
.....H6
.....I6
.....J6
.....K6
.....L6
.....M6
.....N6

.....B7
.....C7