```
                        .REM    &


                                        IDENTIFICATION
                                        --------------

                        PRODUCT CODE:   AC-T587B-MC
                        PRODUCT NAME:   CVMSBB0 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 RESPONSIBLITY 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         PDP             UNIBUS          MASSBUS
                        DEC             DECUS           DECTAPE         DECX/11
```

TABLE OF CONTENTS

```
                        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 WORK-
                        ING 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 MEM-
                        ORY 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.
```

```
                        4.0 ERROR REPORTING

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

                        WHEN AN ERROR OCCURS A "CONTROL &" CAN BE INPUT TO RE-
                        CIEVE EXPANDED ERROR INFORMATION.


                        &



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

                                .ENABL  LC              ;ENABLE LOWER CASE.
000000                          .ENABL  ABS             ;And absolute locations.
```

EQUATES

```
                        .SBTTL  EQUATES


                        ;*****************************************************************************
                        ;Memory Management Equates
                        ;

                                ;+++
                                ;Status Register addresses
                                ;---

177572                          MMUSR0= 177572
172516                          MMUSR3= 172516


                                ;+++
                                ;Kernal Mode PDR addresses
                                ;---

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


                                ;+++
                                ;Kernal Mode PAR addresses
                                ;---

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

EQUATES

```
                        ;***
                        ;Memory Management register settings
                        ;---

004406                  RW256=  4406            ;PDR setting for Read/Write, 320 words.
077406                  RW4096= 77406           ;PDR setting for Read/Write, 4096 words.
077402                  RO4096= 77402           ;PDR setting for Read/Only, 4096 words.
077400                  NR4096= 77400           ;PDR setting for non-resident.

000000                  PAR0K=  0               ;PAR setting for a base address of 0.
000200                  PAR4K=  200             ;PAR setting for a base address of 4K.
000400                  PAR8K=  400             ;PAR setting for a base address of 8K.
000600                  PAR12K= 600             ;PAR setting for a base address of 12K.
001000                  PAR16K= 1000            ;PAR setting for a base address of 16K.
001200                  PAR20K= 1200            ;PAR setting for a base address of 20K.
001400                  PAR24K= 1400            ;PAR setting for a base address of 24K.
177600                  PARIO=  177600          ;PAR setting for the I/O page.

000001                  MMUENA= 1               ;SR0 setting to enable memory management.
000020                  MMU22A= 20              ;SR3 setting to enable 22 bit mapping.

                ;*******************************************************************************
                ;Processor Status Word and Memory time out vector
                ;

                        PSW=    177776
062400                  TIMSCA= 62400
000004                  TIMOUT= 4

                ;*******************************************************************************
                ;Write Package Equates
                ;

000015                  CR=     15              ;ASCII Return.
000012                  LF=     12              ;ASCII Line Feed.
000021                  XON=    21              ;ASCII XON.
000023                  XOFF=   23              ;ASCII XOFF.
000033                  ESC=    33              ;ASCII Escape character.
000007                  BELL=   7               ;ASCII Bell character.
177560                  DEFCON= 177560          ;Default console CSR address.

                ;*******************************************************************************
                ;Equates that allow access to the stack frame.  These are the offsets into
                ;the stack for registers saved during the "Preserve Registers" routine.

000014                  R5SLOT==        14      ;Offset for R5.
000012                  R4SLOT==        12      ;Offset for R4.
000010                  R3SLOT==        10      ;Offset for R3.
000006                  R2SLOT==        6       ;Offset for R2.
000004                  R1SLOT==        4       ;Offset for R1.
000002                  R0SLOT==        2       ;Offset for R0.
```

PROGRAM MACROS

```
                        .SBTTL  PROGRAM MACROS

                                ;+++
                                ;The "PRESERVE" macro facilitates calling the standard
                                ;register preservation routine (PREGO5).
                                ;
                                .macro  PRESERVE
                                JSR     R5,PREGO5
                                .endm   PRESERVE


                                ;+++
                                ;The "PUSH" macro facilitates pushing information on the stack.  Up to
                                ;six items may be placed on the stack with one macro.
                                ;
                                .macro  PUSH    A,B,C,D,E,F
                                .if nb A
                                MOV     A,-(SP)
                                .endc
                                .if nb B
                                MOV     B,-(SP)
                                .endc
                                .if nb C
                                MOV     C,-(SP)
                                .endc
                                .if nb D
                                MOV     D,-(SP)
                                .endc
                                .if nb E
                                MOV     E,-(SP)
                                .endc
                                .if nb F
                                MOV     F,(SP)
                                .endc
                                .endm   PUSH
```

```
                        ;+++
                        ;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
```

```
                        ;+++
                        ;The "RETURN" macro facilitates returning from a subroutine.
                        ;If parameters are passed back, the "PRESERVE" macro MUST be
                        ;used at the start of the subroutine to set up the stack frame.
                        ;Up to six parameters may be passed back.  The first is passed
                        ;in R0, the second in R1 etc.
                        ;---
                        .macro  RETURN  A,B,C,D,E,F
                        .if nb A
                        MOV     A,R0SLOT(SP)
                        .endc
                        .if nb B
                        MOV     B,R1SLOT(SP)
                        .endc
                        .if nb C
                        MOV     C,R2SLOT(SP)
                        .endc
                        .if nb D
                        MOV     D,R3SLOT(SP)
                        .endc
                        .if nb E
                        MOV     E,R4SLOT(SP)
                        .endc
                        .if nb F
                        MOV     F,R5SLOT(SP)
                        .endc
                        RTS     PC
                        .endm   RETURN
```

STARTUP

```
                                .SBTTL  Startup
                        .=      200
000200  000167  000774          JMP     MAIN            ;This enables XXDP+ to start the program
                                                        ;and disables loading the supervisor.
                        .=      1000
```

```
                                .SBTTL  PTYTBL - (Data section) Parity CSR table

                                        ;The parity CSR table is located here in order to get it into hte
                                        ;Read/Write section.

001000                          PTYTBL: .BLKW   33.             ;Maximum of 16 CSRs, two words per CSR
                                                                ;plus one word for a terminator.

001102  000000                  CHBUF:  .WORD   0               ;Will contain any character received by KBDINT.

001104  000000                  UFDCHR: .WORD   0               ;SAVE FOR INPUT CHARA   ;UFD 02/08/84
001106  000000                  GETOUT: .WORD   0               ;GET OUT FLAG           ;UFD 02/08/84
001110  000000                  CCFLAG: .WORD   0               ;CONTROL C FLAG         ;UFD 02/08/84

                                        ;***
                                        ;This is the end of the section that is Read/Write during testing.
                                        ;---

        001200                          .=      1200
```

```
                                .SBTTL MAIN    - 11/23 Plus - Memory Test, main flow
                                ;*************************************************************************
                                ;INPUTS:        [II1]   Processor is in a Reset state.
                                ;               [II2] - From CTRLX only, SWSTAC (in XXDP+) is altered.
                                ;               [II3] - Absolute location 42 has XXDP+ return/Chain Mode flag.
                                ;
                                ;OUTPUTS:       [DO1] - The XXDP+ return/Chain Mode flag in XXRTN.
                                ;               [DO2] - The address of SWSTAC (in XXDP+) in SWADD.
                                ;
                                ;CALLING SEQUENCE:      [EN1] - Standard entry from XXDP+.
                                ;                       [*EN2] - JMP    MAIN2
                                ;                       [*EN3] - Standard restart from XXDP+.
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-EMT 43 (to XXDP+)
                                ;                               [CL2]-WRITE     [CL3]-TEST      [CL4]-WRITLN.
                                ;*************************************************************************

001200                          START:
001200  000240                 MAIN:   NOP                     ;                               ;UFD 02/09/84
001202  005767  004724                 TST     SAVSP           ;FIRST TIME THRUOGH?            ;UFD 02/09/84
001206  001074                         BNE     MAIN1           ;NO SO BRANCH                   ;UFD 02/09/84
001210  010667  004716                 MOV     SP,SAVSP        ;SAVE STACK POINTER             ;UFD 02/09/84
001214  013767  000042  004672         MOV     @#42,XXRTN      ;Save the XXDP+ return/Chain Mode flag in XXRTN
001222  013767  000030  004674         MOV     @#30,SAV30      ;Save the EMT vector
001230  013767  000032  004670         MOV     @#32,SAV32      ;
001236  013767  000052  004664         MOV     @#52,SAV52      ;SAVE LOC 52                    ;UFD 02/09/84
001244  032737  000040  000052         BIT     #40,@#52        ;Are we under UFD CHAIN MODE?
001252  001415                         BEQ     3$              ;No,then continue testing

                                ;+
                                ; IF WE GET HERE WE ARE IN UFD CAHIN MODE
                                ;-

001254  104042                         EMT     42              ;Get DSRERR address
001256  005060  000042                 CLR     42(R0)          ;Initialize DSRERR to no error
001262  012767  177777  004630         MOV     #-1,UFDCHN      ;Flag that we are under UFD CHAIN MODE
001270  032767  000100  004632         BIT     #100,SAV52      ;IN QUIET MODE?                 ;UFD 02/09/84
001276  001403                         BEQ     3$              ;BR IF NOT                      ;UFD 02/09/84
001300  012767  177777  004614         MOV     #-1,UFDQUI      ;SET QUIET FLAG                 ;UFD 02/09/84

001306  105737  000052         3$:     TSTB    @#52            ;IS THIS THE UFD MONITOR ?      ;UFD 04/03/84
001312  100002                         BPL     1$              ;BR IF NOT                      ;UFD 04/03/84
001314  104043                         EMT     +43             ;Get the address of SWSTAC from XXDP+.
001316  000403                         BR      2$              ;Go save it
001320  104042                 1$:     EMT     +42             ;Get the address of dca from monitor
001322  162700  001424                 SUB     #1424,R0        ;Adjust it to SWSTAK            ;UFD 02/09/84
001326  010067  004560         2$:     MOV     R0,SWADD        ;Save it in SWADD.

001332                          CHKCC:  CALL    GETCHR          ;SEE IF A CHARA BEEN TYPED      ;UFD 02/09/84
001332  004767  000546                 JSR     PC,GETCHR
001336  010037  001104                 MOV     R0,@#UFDCHR
001342  022700  000003                 CMP     #3,R0           ;IS IT CONTROL C ?              ;UFD 02/09/84
001346  001006                         BNE     1$              ;NO SO BRANCH                   ;UFD 02/09/84
001350  005237  001110                 INC     @#CCFLAG        ;SET ^C FLAG                    ;UFD 02/09/84
001354  005237  001106         2$:     INC     @#GSTOUT        ;GET GET OUT FLAG               ;UFD 02/09/84
001360                                  CALL    CTRLX           ;YES                            ;UFD 02/09/84
001360  004767  000702                 JSR     PC,CTRLX
```

MAIN - 11/23 PLUS MEMORY TEST, MAIN FLOW

```
                                        1$:     TST     UFDCHN          ;IN UFD CHAIN MODE?                     ;UFD 02/09/84
                                                BEQ     MAIN1           ;NO SO BRANCH                           ;UFD 02/09/84
                                                CMP     #32,R0          ;IS IT CONTROL Z ?                      ;UFD 02/09/84
                                                BEQ     2$              ;YES                                    ;UFD 02/09/84
                                        ;+
                                        ; WE MUST TEST FOR AN ORION PROCESSOR. IF WE HAVE ONE THEN TURN OFF CACHE.
                                        ;-
                                        MAIN1:
                                                MOV     @#10, (SP)      ;SAVE LOC 10                            ;UFD 04/03/84
                                                MOV     #100$,@#10      ;SET FOR NONEX INST TRAP                ;UFD 04/03/84
                                                MFPT                    ;GET PROCESSOR TYPE                     ;UFD 04/03/84
                                                CMP     #5,R0           ;IS IT AN ORION ?                       ;UFD 04/03/84
                                                BNE     101$            ;BR IF NOT                              ;UFD 04/03/84
                                                MOV     #-1,ORIONF      ;SET FLAG                               ;UFD 04/03/84
                                                MOV     @#177746,SAVCAC ;SAVE CACHE                             ;UFD 04/03/84
                                                MOV     #1000,@#177746  ;TURN OFF CACHE                         ;UFD 04/03/84
                                                BR      101$            ;LEAVE NOW                              ;UFD 04/03/84
                                        100$:   MOV     #101$,(SP)      ;WE TRAPPED SO SET UP RTN               ;UFD 04/03/84
                                                RTI                     ;CLEAR STACK FROM TRAP                  ;UFD 04/03/84
                                        101$:   MOV     (SP)+,@#10      ;RESTORE LOC 10                         ;UFD 04/03/84
                                                MOV     PC,SP           ;Initialize the stack pointer.
                                                BIC     #17777,SP       ;
                                                ADD     #1000,SP        ;
                                                CALL    WRITE,#MTINPG   ;Display the "test in progress" message.
                                                MOV     #MTINPG,R5
                                                JSR     PC,WRITE
                                                CALL    TEST            ;Do all the testing.
                                                JSR     PC,TEST
                                                CALL    WRITE,#MTOK     ;Display the "OK" message.
                                                MOV     #MTOK,R5
                                                JSR     PC,WRITE
                                        MAIN2:  TST     ORIONF          ;ORION CP ?                             ;UFD 04/03/84
                                                BEQ     9$              ;BR IF NOT                              ;UFD 04/03/84
                                                MOV     SAVCAC,@#177746 ;RESTORE CACHE                          ;UFD 04/03/84
                                        9$:
                                                MOV     SAV52,@#52      ;RESTORE LOC 52                         ;UFD 02/09/84
                                                MOV     SAV30,@#30      ;RESTORE 30                             ;UFD 02/09/84
                                                MOV     SAV32,@#32      ;RESTORE 32                             ;UFD 02/09/84
                                                MOV     XXRTN,@#42      ;Restore location 42.
                                                BEQ     2$              ;If not in chain mode, loop.
                                                TST     @#CCFLAG        ;^C TYPED ?                             ;UFD 02/09/84
                                                BEQ     1$              ;NO, SO BRANCH                          ;UFD 02/09/84
                                                TSTB    @#52            ;IS THIS THE UFD MONITOR ?              ;UFD 04/03/84
                                                BPL     1$              ;BR IF NOT                              ;UFD 04/03/84
                                                EMT     42              ;GET ADDR OF UCA                        ;UFD 02/09/84
                                                MOV     #1,46(R0)       ;SET PASS CNT TO 1                      ;UFD 02/09/84
```

```
    503 001610  004777  004300          1$:     JSR     PC,@XXRTN       ;Return to XXDP+ CHAIN FILE
    504 001614  000406                          BR      4$              ;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 POINTER           ;UFD 02/09/84
    509 001630  000207                          RTS     PC              ;RETURN TO XXDP+ MONITOR        ;UFD 02/09/84
    510
    511 001632                          4$:     CALL    WRITE,@NEWLIN   ;Do a return and line feed for multiple passes.
        001632  012705  000001                  MOV     @NEWLIN,R5
        001636  004767  003742                  JSR     PC,WRITE
    512 001642  000656                          BR      MAIN1           ;Loop if multiple Chain Mode passes.
```

```
                        .SBTTL KTINT  - Handle Memory Management traps
                        ;******************************************************************
                        ;INPUTS:        [DI1]   TKTERR, the address of the error message table.
                        ;
                        ;OUTPUTS:       [AD01] - KT trap error message to the console.
                        ;
                        ;CALLING SEQUENCE:      [*AV2] - Trap through vector 250.
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-ERROR.
                        ;******************************************************************

001644  004736          KTINT:  JSR     PC,@(SP)+       ;Routine locater.

001646                          PUSH    R5              ;Save the parameter passing register.
001646  010546                  MOV     R5,-(SP)

001650  004767  005146          JSR     PC,TKTERR       ;Get the address of the error message table.
001654                          CALL    ERRCR,(SP)+     ;Signal the error using the table address.
001654  012605                  MOV     (SP)+,R5
001656  004767  003562          JSR     PC,ERROR

001662  012737  000001  177572  MOV     @MMUENA,@#MMUSR0 ;Reset Memory management.

001670                          POP     R5              ;Restore the register.
001670  012605                  MOV     (SP)+,R5
001672  000002                  RTI                     ;Return from interrupt.
```

```
                        .SBTTL PTYINT- Handle Parity traps
                        ;*****************************************************************************
                        ;INPUTS:        [DI1]   -The address of the parity CSR table (PTYTBL).
                        ;
                        ;OUTPUTS:       [DO1]   -The CSR which caused the trap is reset for write
                        ;                         good parity, traps enabled.
                        ;               [DO2]   -The error count associated with the CSR is
                        ;                         incremented.
                        ;
                        ;CALLING SEQUENCE:      [*AV2] (Trap through vector 114)
                        ;
                        ;COMMENTS:      If no CSR can be found which caused a trap, exit to UNXINT,
                        ;               to signal an unexpected interrupt.
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [*CL1] - UNXINT.
                        ;*****************************************************************************

001674  004736          PTYINT: JSR     PC,@(SP)+       ;Routine locater.

001676                          PUSH    R0              ;Save a working register.
001676  010046                  MOV     R0,-(SP)

001700  012700  001000          MOV     #PTYTBL,R0      ;Address of PTYTBL in the working register.
001704  000401                  BR      2$              ;

001706  005720          1$:     TST     (R0)+           ;Get past the error count.

001710  005710          2$:     TST     (R0)            ;Check for the table terminator.
001712  001003                  BNE     3$              ;
001714                          POP     R0              ;If terminator found, restore R0
001714  012600                  MOV     (SP)+,R0
001716  000167  004614          JMP     VCT114          ;And go signal an unexpected trap.

001722  005730          3$:     TST     @(R0)+          ;Check the CSR for a parity error.
001724  100370                  BPL     1$              ;

001726  005210                  INC     (R0)            ;Bump the error count.
001730  012750  000001          MOV     #1,@-(R0)       ;Reset the Parity CSR.

001734                          POP     R0              ;Restore the working register.
001734  012600                  MOV     (SP)+,R0
001736  000002                  RTI                     ;Return from trap.
```

UNXINT - HANDLE UNEXPECTED INTERRUPTS.

```
                                .SBTTL UNXINT- Handle unexpected interrupts.
                                ;*****************************************************************************
                                ;INPUTS:        [DI1]   The address of the vectors table.
                                ;
                                ;OUTPUTS:       [ADO1] - Error message to the console.
                                ;
                                ;CALLING SEQUENCE:      [*AV3] Trap through any unused vector.
                                ;
                                ;COMMENTS:      UNXERR does the error handling for this routine.
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-UNXERR.
                                ;*****************************************************************************

001740  004767  004454          UNXINT: JSR     PC,VCTORS       ;Get the address of the vector table.

001744  162616                          SUB     (SP)+,(SP)      ;
001746  162716  000004                  SUB     #4,(SP)         ;Form the address of the trapping vector.

001752  011646                          MOV     (SP),-(SP)      ;Form a stack frame and save R5.
001754  010566  000002                  MOV     R5,2(SP)        ;

001760                                  CALL    UNXERR,(SP)+    ;Pass the vector address to the error handler.
001760  012605                          MOV     (SP)+,R5
001762  004767  002700                  JSR     PC,UNXERR

001766                                  POP     R5              ;Restore R5.
001766  012605                          MOV     (SP)+,R5

001770  000002                  RTI     ;Return from interrupt.
```

KBDINT - HANDLE CONSOLE KEYBOARD INTERRUPTS.

```
                                .SBTTL KBDINT - Handle console keyboard interrupts.
                                ;*****************************************************************************
                                ;INPUTS:        [DI1]   The input character.
                                ;
                                ;OUTPUTS:       [DO1] - The character is placed in CHBUF.
                                ;
                                ;CALLING SEQUENCE:      [*AV4] - Interrupt through vector 60.
                                ;
                                ;COMMENTS:      If the input character is a Control X, exit the program via
                                ;               CTRLX.
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [*CL1]-CTRLX.
                                ;*****************************************************************************

001772  004736                  KBDINT: JSR     PC,@(SP)+       ;Routine locater.

001774  000240                          NOP                                             ;UFD 02/08/84
001776  017737  004140  001102          MOV     @#BBUF,@#CHBUF  ;Put the character in CHBUF.
002004  042737  177600  001102          BIC     #177600,@#CHBUF ;Strip the character to 7 bits.
002012  005767  004104                  TST     UFDQUI          ;IS IT UFD QUIET MODE?  ;UFD 02/08/84
002016  001004                          BNE     2$              ;DON'T TEST ^X IF IT IS ;UFD 02/08/84
002020  022737  000030  001102          CMP     #30,@#CHBUF     ;See if the character is Control X.
002026  001420                          BEQ     6$              ;IT IS SO GOTO CTRLX    ;UFD 02/08/84

002030  022737  000003  001102  2$:     CMP     #3,@#CHBUF      ;IS IT ^C ?             ;UFD 02/09/84
002036  001410                          BEQ     4$              ;YES, SO BRANCH         ;UFD 02/08/84
002040  005767  004054                  TST     UFDCHN          ;UFD CHAIN MODE ?       ;UFD 02/08/84
002044  001416                          BEQ     8$              ;LEAVE NOW IF NOT       ;UFD 02/08/84
002046  022737  000032  001102          CMP     #32,@#CHBUF     ;IS IT A ^Z ?           ;UFD 02/08/84
002054  001403                          BEQ     5$              ;YES, SO LEAVE          ;UFD 02/08/84
002056  000411                          BR      8$              ;IGNORE CHARACTER       ;UFD 02/08/84

002060  005237  001110          4$:     INC     @#CCFLAG        ;SET ^C FLAG            ;UFD 02/08/84
002064  005237  001106          5$:     INC     @#GETOUT        ;SET GET OUT FLAG       ;UFD 02/08/84
002070  013737  001102  001104  6$:     MOV     @#CHBUF,@#UFDCHR ;SAVE IT               ;UFD 02/08/84
002076                                  CALL    CTRLX           ;If it is, Goto CTRLX.
        002076  004767  000164          JSR     PC,CTRLX

002102  000002                  8$:     RTI                     ;Return from interrupt.
```

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

```
                                .SBTTL GETCHR - (Write Package) Get a single character input from the console
                                ;*****************************************************************************
                                ;INPUTS:        [DI1]   CHBUF.
                                ;               [*DI2] - The input character.
                                ;
                                ;OUTPUTS:       [OE1] - R0 contains the character, stripped to 7 bits.  If no
                                ;                       character is received, R0 is returned zero.
                                ;
                                ;CALLING SEQUENCE:      CALL    GETCHR
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   None.
                                ;*****************************************************************************

002104  013700  001102          GETCHR: MOV     @#CHBUF,R0      ;Check the character buffer.
002110  001403                          BEQ     1$              ;
                                
002112  005037  001102                  CLR     @#CHBUF         ;If a character was in CHBUF, clear CHBUF.
002116  000405                          BR      2$              ;And exit.
                                
002120  105777  004014          1$:     TSTB    @#KBCSR         ;Check for inputs.
002124  100002                          BPL     2$              ;If none, get out.
                                
002126  117700  004010                  MOVB    @#KBBUF,R0      ;If input, get it.
                                
002132  042700  177600          2$:     BIC     #177600,R0      ;Strip to 7 bits.
                                
002136                                  RETURN
002136  000207                          RTS     PC
```

```
                                .SBTTL PRTCHR - (WRITE PACKAGE) Display a single character on the console
                                ;*****************************************************************************
                                ;INPUTS:        [IE1]   R5 contains the character to be displayed.
                                ;
                                ;OUTPUTS:       [DO1] - The character is displayed on the console.
                                ;
                                ;COMMENTS:      If an XOFF is received from the console terminal, this
                                ;               routine waits for an XON before returning.
                                ;
                                ;CALLING SEQUENCE:      CALL    PRTCHR,IE1
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1],[*CL2]-GETCHR.
                                ;*****************************************************************************

002140  005767  003756          PRTCHR: TST     UFDQUI          ;Are we running under UFD ?
002144  001401                          BEQ     FPRTCH          ;Yes,then skip the typeout
002146                                  RETURN                                  ;UFD 02/08/84
002146  000207                          RTS     PC

002150                          FPRTCH: PUSH    R0              ;Save R0.
002150  010046                          MOV     R0,-(SP)

002152  105777  003766          1$:     TSTB    @DSPCSR         ;Check the transmit ready bit on tne console.
002156  100375                          BPL     1$              ;If not ready, wait for it.
002160  110577  003762                  MOVB    R5,@DSPBUF      ;Send the character.

002164                                  CALL    GETCHR          ;Get an input character.
00216.  004767  177714                  JSR     PC,GETCHR
002170  022700  000023                  CMP     #XOFF,R0        ;See if it is an XOFF.
002174  001005                          BNE     3$              ;If not, ignore it and get out.

002176                          2$:                             ;If XOFF, wait for an XON.
002176                                  CALL    GETCHR          ;Get a character.
002176  004767  177702                  JSR     PC,GETCHR
002202  022700  000021                  CMP     #XON,R0         ;See if it is an XON.
002206  001373                          BNE     2$              ;If not, loop and wait.
                                                                ;If XON, return.

002210                          3$:     POP     R0              ;Restore R0.
002210  012600                          MOV     (SP)+,R0
002212                                  RETURN
002212  000207                          RTS     PC
```

TEST TEST ALL OF MAIN MEMORY.

```
                                        .SBTTL TEST   - Test all of main memory.
                                        ;*****************************************************************************
                                        ;INPUTS:        [II1]   The processor is in a Reset state.
                                        ;
                                        ;OUTPUTS:       [AD01] - Memory is sized.  The size and test time is displayed.
                                        ;               [AD02] - Memory is tested.  Any errors are displayed on the
                                        ;                        console.
                                        ;
                                        ;CALLING SEQUENCE:      CALL    TEST
                                        ;
                                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-MAPINI    [CL2]-SIZMEM    [CL3]-TPARTY
                                        ;                               [CL4]-TL16K     [CL5]-TREST     [CL6]-CPARTY
                                        ;*****************************************************************************
                                        
002214  000240                  TEST:   NOP                     ;                               ;UFD 03/13/84
002216                                  CALL    MAPINI          ;Set initial conditions for testing.
002216  004767  000250                  JSR     PC,MAPINI
002222  000240                          NOP                     ;                               ;UFD 03/13/84
002224                                  CALL    SIZMEM          ;Size the memory and get back the highest
002224  004767  000420                  JSR     PC,SIZMEM
002230  000240                          NOP                     ;responding PAR setting.        ;UFD 03/13/84
002232                                  CALL    TPARTY,R0       ;Test parity detect and enable parity traps.
002232  010005                          MOV     R0,R5
002236  004767  000750                  JSR     PC,TPARTY
002240  000240                          NOP                     ;Pass the highest PAR setting.  ;UFD 03/13/84
002242                                  CALL    TL16K           ;Test the low 16K words of memory.
002242  004767  001172                  JSR     PC,TL16K
002246  000240                          NOP                     ;                               ;UFD 03/13/84
002250                                  CALL    TREST           ;Test whatever is left.
002250  004767  001212                  JSR     PC,TREST
002254  000240                          NOP                     ;Pass the highest PAR setting.  ;UFD 03/13/84
002256                                  CALL    CPARTY          ;Check for any parity errors during testing.
002256  004767  001530                  JSR     PC,CPARTY
002262  000240                          NOP                     ;                               ;UFD 03/13/84
002264                                  RETURN                  ;Return.
002264  000207                          RTS     PC
```

```
                                        .SBTTL CTRLX  - Handle a Control X keyboard input
                                        ;*****************************************************************************
                                        ;INPUTS:        [DI1]   SWADD.
                                        ;
                                        ;OUTPUTS:       [DO1] - SWSTAC (in XXDP+) is altered to disable Chain Mode.
                                        ;
                                        ;CALLING SEQUENCE:      CALL    CTRLX
                                        ;
                                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-SETBAK    [CL2]-MAIN2.
                                        ;*****************************************************************************

002266  013701  001102          CTRLX:  MOV     @#CHBUF,R1      ;SAVE LOCATION          ;UFD 02/10/84
002272  013702  001104                  MOV     @#UFDCHR,R2     ;SAVE LOCATION          ;UFD 02/10/84
002276  013703  001106                  MOV     @#GETOUT,R3     ;SAVE LOCATION          ;UFD 02/10/84
002302  013704  001110                  MOV     @#CCFLAG,R4     ;SAVE LOCATION          ;UFD 02/10/84

002306                                  CALL    SETBAK          ;Restore original program location and reset.
002306  004767  001742                  JSR     PC,SETBAK

002312  010137  001102                  MOV     R1,@#CHBUF      ;RESTORE LOCATION       ;UFD 02/10/84
002316  010237  001104                  MOV     R2,@#UFDCHR     ;RESTORE LOCATION       ;UFD 02/10/84
002322  010337  001106                  MOV     R3,@#GETOUT     ;RESTORE LOCATION       ;UFD 02/10/84
002326  010437  001110                  MOV     R4,@#CCFLAG     ;RESTORE LOCATION       ;UFD 02/10/84
002332  000240                          NOP
002334                                  CALL    FPRTCH,#'↑      ;PRINT UPARROW          ;UFD 02/08/84
002334  012705  000136                  MOV     #'↑,R5
002340  004767  177604                  JSR     PC,FPRTCH
002344  062737  000100  001104          ADD     #100,@#UFDCHR   ;MAKE CHARA ASCII       ;UFD 02/08/84
002352                                  CALL    FPRTCH,@#UFDCHR ;PRINT CHARACTER        ;UFD 02/08/84
002352  013705  001104                  MOV     @#UFDCHR,R5
002356  004767  177566                  JSR     PC,FPRTCH
002362                                  CALL    FPRTCH,#40      ;PRINT A SPACE          ;UFD 02/08/84
002362  012705  000040                  MOV     #40,R5
002366  004767  177556                  JSR     PC,FPRTCH

002372  016700  003514                  MOV     SWADD,R0        ;Get the address of SWSTAC in XXDP+.
002376  005767  003516                  TST     UFDCHN          ;UFD CHAIN MODE ?       ;UFD 02/08/84
002402  001014                          BNE     2$              ;YES, SO BRANCH         ;UFD 02/08/84
002404  022737  000130  001104          CMP     #'X,@#UFDCHR    ;IS IT AN X ?           ;UFD 02/08/84
002412  001025                          BNE     8$              ;LEAVE IF NOT           ;UFD 02/08/84

002414  105710                  1$:     TSTB    (R0)            ;Check for the stack terminator.
002416  001423                          BEQ     8$              ;If found, get out.

002420  122720  000101                  CMPB    #'A,(R0)+       ;Find the auto mode switch.
002424  001373                          BNE     1$              ;
002426  112740  000130                  MOVB    #'X,-(R0)       ;Replace it with the exit switch.
002432  000415                          BR      8$              ;UFD 02/08/84

002434  005737  001110          2$:     TST     @#CCFLAG        ;IS IT UFD AND ^C ?     ;UFD 02/08/84
002440  001412                          BEQ     8$              ;NO, SO BRANCH          ;UFD 02/08/84
002442  105720                  3$:     TSTB    (R0)+           ;FIND END OF STACK      ;UFD 02/08/84
002444  001376                          BNE     3$
002446  112760  000057  177777          MOVB    #'/,-1(R0)      ;LOAD SLASH             ;UFD 02/08/84
002454  112720  000136                  MOVB    #'↑,(R0)+       ;LOAD UPARROW           ;UFD 02/08/84
002460  112720  000103                  MOVB    #'C,(R0)+       ;LOAD C                 ;UFD 02/08/84
002464  105010                          CLRB    (R0)            ;SET END OF STACK       ;UFD 02/08/84
```

```
 787
 788 002466  004767  177024          8$:     JSR     PC,MAIN2        ;Goto MAIN (EN2).
```

MAPINI - SET INITIAL PROGRAM CONDITIONS

```
                                .SBTTL MAPINI - Set initial program conditions
                                ;*****************************************************************************
                                ;INPUTS:        [II1]   SWADD is set up with the address of SWSTAC in XXDP+.
                                ;
                                ;OUTPUTS:       [*D01] - XXRTN is cleared if XXDP+ is incorrectly located.
                                ;               [D02]  - The SP is adjusted for virtual memory 1000 and down.
                                ;               [D03]  - Keyboard interrupts are enabled.
                                ;               [D04]  - The virtual vector area is loaded with vectors.
                                ;               [AD01] - The program is in 16K to 20K area, the next 4K is
                                ;                        tested.
                                ;               [AD02] - Memory Management is mapped and enabled,
                                ;                        priority zero is set.
                                ;
                                ;CALLING SEQUENCE:      CALL    MAPINI
                                ;
                                ;COMMENTS:      All registers are corrupted by this routine.
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [*CL1]-ERROR    [CL2]-BUMP      [CL3]-SETABL
                                ;*****************************************************************************
002472  016700  003414          MAPINI: MOV     SWADD,R0        ;Get the address of SWSTAC (in XXDP+) for
002476  042700  007777                  BIC     #7777,R0        ;Strip unused bits.
002502  022700  150000                  CMP     #150000,R0      ;Check for correct location.
002506  001407                          BEQ     1$              ;
002510  004767  004330                  JSR     PC,TLOW32       ;If incorrect, get the message table address.
002514                                  CALL    ERROR,(SP)+     ;Do the error handling.
002514  012605                          MOV     (SP)+,R5
002516  004767  002722                  JSR     PC,ERROR
002522  005067  003366                  CLR     XXRTN           ;Disable return to XXDP+
                                
002526                          1$:     CALL    BUMP            ;Ensure the program is in the 16K area,
002526  004767  000772                  JSR     PC,BUMP
                                                                ;test the segment address table area.
002532  004767  003662                  JSR     PC,VCTORS       ;Get the address of the vectors table.
002536  012600                          MOV     (SP)+,R0        ;
002540  012701  100000                  MOV     #100000,R1      ;Set R1 to address the current 4K.
002544  012702  000100                  MOV     #64.,R2         ;Set a counter to load the vector area.

002550  010021                  2$:     MOV     R0,(R1)+        ;Load a vector.
002552  062700  000004                  ADD     #4,R0           ;Set up the next address.
002556  012721  000340                  MOV     #340,(R1)+      ;
002562  077206                          SOB     R2,2$           ;Until all 64 are loaded.

002564  004767  177054                  JSR     PC,KTINT        ;Get the address of the KT trap handler.
002570  012637  100250                  MOV     (SP)+,@#100250  ;Set up the KT trap vector.
002574  004767  177074                  JSR     PC,PTYINT       ;Get the address of the parity trap handler.
002600  012637  100114                  MOV     (SP)+,@#100114  ;Set up the parity trap vector.
002604  004767  177162                  JSR     PC,KBDINT       ;Get the address of the keyboard interrupt
002610  012637  100060                  MOV     (SP)+,@#100060  ;handler and set up the keyboard vector.

002614  004767  003330                  JSR     PC,TKMAP1       ;Get the address of the initial map.
002620                                  CALL    SETABL,(SP)+    ;Go set it up.
002620  012605                          MOV     (SP)+,R5
002622  004767  001520                  JSR     PC,SETABL

002626  012767  100000  003262          MOV     #100000,RELFLG  ;Set RELFLG to reset the SP, don't relocate.
002634  042706  160000                  BIC     #160000,SP      ;Adjust the stack pointer.
002640  012777  000100  003272          MOV     #100,@KBCSR     ;Enable keyboard interrupts.
```

```
842
843 002646                              RETURN
    002646  000207                      RTS     PC
```

```
                        .SBTTL SIZMEM - Determine the memory size
                        ;*************************************************************************
                        ;INPUTS:        [II1]   Memory management is set up and enabled (22 bit).
                        ;
                        ;OUTPUTS:       [OE1] - The highest valid (responding) PAR setting.
                        ;               [ADO1] - The results of the sizing are displayed.
                        ;
                        ;CALLING SEQUENCE:      CALL    SIZMEM
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-DSPSIZ
                        ;*************************************************************************

002650  000240         SIZMEM: NOP                             ;UFD 03/13/84
002652                          PRESERVE                        ;Preserve registers.
002652  004567  003174          JSR     R5,PREGOS
002656  013746  000004          MOV     @#TIMOUT,-(SP)  ;Save the current timeout vector.
002662  004767  000072          JSR     PC,2$           ;Get the address of the local timeout handler.
002666  012637  000004          MOV     (SP)+,@#TIMOUT  ;Put it in the vector location.
002672  012700  177600          MOV     #177600,R0      ;Set R0 to the I/O page PAR setting.
002676  162700  000200  1$:     SUB     #200,R0         ;Set R0 to the next lower PAR setting.
002702  010037  172346          MOV     R0,@#KPAR1      ;Use it to set PAR 1.
002706  012737  000000  037776  MOV     #0,@#037776     ;Address something in PAR 1 range.
002714  001370                  BNE     1$              ;If nothing there, try next lower setting.
002716  012637  000004          MOV     (SP)+,@#TIMOUT  ;Restore the timeout vector.
002722  010002                  MOV     R0,R2           ;Scale the PAR setting into K bytes.
002724  062702  000200          ADD     #200,R2         ;
002730  006202                  ASR     R2              ;Divide by 16.
002732  042702  100000          BIC     #100000,R2      ;
002736  006202                  ASR     R2              ;
002740  006202                  ASR     R2              ;
002742  006202                  ASR     R2              ;

002744                          CALL    DSPSIZ,R2       ;Go display the memory size and test time.
002744  010205                  MOV     R2,R5
002746  004767  000020          JSR     PC,DSPSIZ

002752                          RETURN  R0              ;Return the highest responding PAR setting.
002752  010066  000002          MOV     R0,ROSLOT(SP)
002756  000207                  RTS     PC

002760  004736          2$:     JSR     PC,@(SP)+       ;Locater for the local handler.

002762  042766  000004  000002  BIC     #4,2(SP)        ;Clear the Z flag in the saved PSW.
002770  000002                  RTI                     ;Return from interrupt.
```

DSPSIZ - DISPLAY MEMORY SIZE AND TEST TIME

```
                                .SBTTL DSPSIZ   Display memory size and test time
                                ;*****************************************************************************
                                ;INPUTS:        [IE1]   Memory size in Kbytes.
                                ;
                                ;OUTPUTS:       Size and test time are displayed on the console.
                                ;
                                ;CALLING SEQUENCE:      CALL    DSPSIZ,IE1
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-WRIUSN    [CL2]-WRITE     [CL3]-WRIUSN
                                ;                               [*CL4]-WRITE    [CL5]-WRIUSN    [*CL6]-WRITE
                                ;                               [CL7]-PRTCHR
                                ;*****************************************************************************

002772  000240                  DSPSIZ: NOP                             ;UFD 03/13/84
002774                                  PRESERVE                        ;Preserve registers.
002774  004567  003052                  JSR     R5,PREG05
003000  010502                          MOV     R5,R2                   ;Put size in Kbytes in R2.
003002  012746  000041                  MOV     #33.,-(SP)              ;Initialize character count on the stack.

003006                                  CALL    WRIUSN,R2,#0,#10.       ;Display the memory size in K bytes.
003006  010205                          MOV     R2,R5
003010  012704  000000                  MOV     #0,R4
003014  012703  000012                  MOV     #10.,R3
003020  004767  002614                  JSR     PC,WRIUSN
003024  060016                          ADD     R0,(SP)                 ;Update the character count.
003026                                  CALL    WRITE,@#TTIME           ;Display the "test time" message.
003026  012705  000020                  MOV     @#TTIME,R5
003032  004767  002546                  JSR     PC,WRITE

003036  070227  062400                  MUL     @#TTMSCA,R2             ;Scale K bytes into test time in seconds.
003042  010203                          MOV     R2,R3                   ;
003044  005002                          CLR     R2                      ;
003046  071227  000074                  DIV     #60.,R2                 ;Convert seconds to minutes and seconds.
003052  010301                          MOV     R3,R1                   ;Save seconds.

003054                                  CALL    WRIUSN,R2,#0,#10.       ;Display test time minutes.
003054  010205                          MOV     R2,R5
003056  012704  000000                  MOV     #0,R4
003062  012703  000012                  MOV     #10.,R3
003066  004767  002546                  JSR     PC,WRIUSN
003072  060016                          ADD     R0,(SP)                 ;Update the character count.

003074  005700                          TST     R0                      ;Check for no minutes.
003076  001406                          BEQ     1$                      ;If no minutes, don't display "minutes".
003100                                  CALL    WRITE,@#TTIMM           ;Display "minutes.
003100  012705  000047                  MOV     @#TTIMM,R5
003104  004767  002474                  JSR     PC,WRITE
003110  062716  000012                  ADD     #10.,(SP)               ;Update the character count.

003114                          1$:     CALL    WRIUSN,R1,#0,#10.       ;Display seconds.
003114  010105                          MOV     R1,R5
003116  012704  000000                  MOV     #0,R4
003122  012703  000012                  MOV     #10.,R3
003126  004767  002506                  JSR     PC,WRIUSN
003132  060016                          ADD     R0,(SP)                 ;Update the character count.

003134  005700                          TST     R0                      ;Check for no seconds.
```

```
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
```

```
                        .SBTTL TPARTY - Enable parity traps and test parity detect
                        ;*****************************************************************************
                        ;INPUTS:        [IE1]   The highest valid PAR setting.
                        ;               [II1] - Memory Management is set up and enabled (22 bit).
                        ;
                        ;OUTPUTS:       [OI1] - PTYTBL, the parity CSR table is established with all
                        ;                       error counts initialized to zero.
                        ;               [OI2] - All live parity CSRs are set for write good parity,
                        ;                       traps enabled.
                        ;
                        ;CALLING SEQUENCE:      CALL    TPARTY,IE1
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-CPARTY    [CL2]-SETABL    [CL3]-CPARTY
                        ;*****************************************************************************
003210  000240          TPARTY: NOP                             ;UFD 03/13/84
003212                          PRESERVE                        ;Preserve registers.
003212  004567  002634          JSR     R5,PREG05
003216  012704  001000          MOV     #PTYTBL,R4              ;Get the address of the parity CSR table.
003222  004767  003772          JSR     PC,WRNGPA               ;Get the address of the "write wrong parity"
                                                                ;table.
003226                          CALL    SPARTY,(SP)+            ;Pass it to SPARTY for setup, set up PTYTBL
003226  012605                  MOV     (SP)+,R5
003230  004767  000422          JSR     PC,SPARTY
                                                                ;with all error counts at -4.
003234                          PUSH    @#0114                  ;Save current contents of the parity vecto
003234  013746  000114          MOV     @#0114,-(SP)
003240  004767  000116          JSR     PC,7$                   ;Get the address of the local trap handler.
003244  012637  000114          MOV     (SP)+,@#0114            ;Set up the parity trap vector.
003250  005001                  CLR     R1                      ;Initialize the parity word.
003252  005002                  CLR     R2                      ;Initialize the byte/word addressing flag.
                                
003254                  1$:     CALL    SETABL                  ;Set all live CSRs for "write wrong parity".
003254  004767  001066          JSR     PC,SETABL
003260  012703  001600          MOV     #1600,R3                ;Set up the first PAR setting.
                                
003264  010337  172342  2$:     MOV     R3,@#KPAR1              ;Set up the working PAR.
003270  005702                  TST     R2                      ;
003272  100403                  BMI     3$                      ;
003274  110137  037777          MOVB    R1,@#37777              ;Write something into the PARs range.
003300  000402                  BR      4$                      ;
003302  010137  037776  3$:     MOV     R1,@#37776              ;
003306  013700  037776  4$:     MOV     @#37776,R0              ;Read it back for possible bad parity.
003312  010137  037776  5$:     MOV     R1,@#37776              ;Write good parity.
003316  013700  037776          MOV     @#37776,R0              ;Read back good parity.
                                
003322  062703  002000          ADD     #2000,R3                ;Bump the PAR setting by 32K words.
003326  020366  000016          CMP     R3,R5SLOT+2(SP)         ;See if the range of memory has been covered.
003332  101754                  BLOS    2$                      ;If not, go do it again.
                                
003334  005101                  COM     R1                      ;Set R1 for the opposite parity.
003336  100746                  BMI     1$                      ;Do both parities.
003340  005702                  TST     R2                      ;
003342  100402                  BMI     6$                      ;If word and byte addressing done, get out.
003344  005102                  COM     R2                      ;Set for word addressing.
003346  000742                  BR      1$                      ;Loop.
```

TPARTY - ENABLE PARITY TRAPS AND TEST PARITY DETECT

```
 997
 998 003350                          6$:   POP     @#114           ;Restore the parity trap vector.
     003350  012637  000114                MOV     (SP)+,@#114
 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
```

```
1002
1003
1004                                            ;+++
1005                                            ;Local parity trap handler.  Traps with write wrong parity set may
1006                                            ;result in wrong parity on the stack.  This routine cleans the stack,
1007                                            ;resets the trapping CSR and increments its associated error count.
1008                                            ;---
1009
1010 003362  004736              7$:    JSR     PC,@(SP)+       ;Locater for the local handler.
1011
1012 003364  010400                     MOV     R4,R0           ;Set R0 to address the parity CSR table.
1013
1014 003366  005710              8$:    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                                                            ;is not setting the parity error bit.  If
1020                                                            ;"write wrong parity" is still enabled for
1021                                                            ;the stack area, the error message may never
1022                                                            ;get displayed and the processor may halt.
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$:    MOV     #1,@(R0)+       ;If it did, reset the CSR.
1031 003424  005210                     INC     (R0)            ;Bump the error count.
1032 003426  010026                     MOV     R0,(SP)+        ;Clean up the stack.
1033 003430  010026                     MOV     R0,(SP)+        ;
1034 003432  005037  177776             CLR     @#PSW           ;Reset Priority zero.
1035 003436  000725                     BR      5$              ;Get back to the test.
```

```
                                        .SBTTL TL16K  - Test the low 16K words of memory
                                        ;*****************************************************************************
                                        ;INPUTS:        [II1]   Memory management is set up and enabled.
                                        ;
                                        ;OUTPUTS:       [AD01] - The low 16K words of memory is tested.
                                        ;
                                        ;CALLING SEQUENCE:      CALL    TL16K
                                        ;
                                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-CLRMEM    [CL2]-TSTMEM
                                        ;*****************************************************************************

003440  000240                  TL16K:  NOP                     ;UFD 03/13/84
003442                                  PRESERVE                ;Preserve registers.
003442  004567  002404                  JSR     R5,PREGOS

003446  005005                          CLR     R5              ;Set the low PAR setting to 0-4K.
003450  012704  000600                  MOV     #600,R4         ;Set the high PAR setting to 12-16K.

003454                                  CALL    CLRMEM          ;Clear the low 16K words.
003454  004767  000370                  JSR     PC,CLRMEM
003460                                  CALL    TSTMEM          ;Test the low 16K words.
003460  004767  000440                  JSR     PC,TSTMEM

003464                                  RETURN
003464  000207                          RTS     PC
```

```
                                .SBTTL TREST  - Test the remainder of memory
                                ;*****************************************************************************
                                ;INPUTS:        [IE1]   The highest valid PAR setting.
                                ;               [II1] - Memory management is set up and enabled (22 bits).
                                ;
                                ;OUTPUTS:       [AD01] - Memory above 16K words is tested.
                                ;
                                ;CALLING SEQUENCE:      CALL    TREST,PE1
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-SETLOW    [CL2]-CLRMEM    [CL3]-TSTMEM
                                ;                               [CL4]-SETBAK
                                ;*****************************************************************************

003466  000240                  TREST:  NOP                     ;UFD 03/13/84
003470                                  PRESERVE                ;Preserve registers.
003470  004567  002356                  JSR     R5,PREGOS

003474  010504                          MOV     R5,R4           ;Set up the high PAR setting.
003476  012705  001000                  MOV     #1000,R5        ;Set up the low PAR setting.

003502                                  CALL    SETLOW          ;Set up for operation in the low 16K words.
003502  004767  000504                  JSR     PC,SETLOW

003506                                  CALL    CLRMEM          ;Clear all memory above 16K words.
003506  004767  000336                  JSR     PC,CLRMEM

003512                                  CALL    TSTMEM          ;Test all that memory.
003512  004767  000406                  JSR     PC,TSTMEM

003516                                  CALL    SETBAK          ;Restore the program to its old location.
003516  004767  000532                  JSR     PC,SETBAK

003522                                  RETURN
003522  000207                          RTS     PC
```

```
                                        .SBTTL BUMP   - Relocate the program up until it resides in 16K to 20K
                                        ;*****************************************************************************
                                        ;INPUTS:        [DI1]   The current program location.
                                        ;
                                        ;OUTPUTS:       [*DO1] - The program is relocated if not already at 16K.
                                        ;               [DO2]  - The 20K to 24K memory segment is tested (quick test)
                                        ;
                                        ;CALLING SEQUENCE:      CALL    BUMP
                                        ;
                                        ;COMMENTS:      All registers are corrupted by this routine,
                                        ;
                                        ;SUBORDINATE ROUTINES CALLED:   [*CL1]-ERROR.
                                        ;*****************************************************************************
003524  010700                  BUMP:   MOV     PC,R0           ;Establish the current program location.
003526  042700  017777                  BIC     #17777,R0       ;
003532  010002                          MOV     R0,R2           ;
003534  010001                          MOV     R0,R1           ;
003536  012705  010000                  MOV     #4096.,R5       ;Set a 4K counter.
003542  005004                          CLR     R4              ;Initialize the program checksum.

003544  062204                  1$:     ADD     (R2)+,R4        ;Calculate the program checksum.
003546  077502                          SOB     R5,1$           ;

003550  012705  010000          2$:     MOV     #4096.,R5       ;Set a 4K counter.
003554  012012                  3$:     MOV     (R0)+,(R2)      ;Copy the program to the next 4K segemnt.
003556  005112                          COM     (R2)            ;
003560  005122                          COM     (R2)+           ;Quick-test all locations.
003562  077504                          SOB     R5,3$           ;

003564  012705  010000                  MOV     #4096.,R5       ;Set a 4K counter.
003570  005003                          CLR     R3              ;Initialize the copy checksum.
003572  062103                  4$:     ADD     (R1)+,R3        ;Calculate the copy checksum.
003574  077502                          SOB     R5,4$           ;

003576  020304                          CMP     R3,R4           ;Compare the copy checksum to the program
003600  001405                          BEQ     5$              ;checksum.
003602  004767  003236                  JSR     PC,TLOW32       ;If they aren't the same, signal an error.
003606                                  CALL    ERROR,(SP)+     ;
003606  012605                          MOV     (SP)+,R5
003610  004767  001630                  JSR     PC,ERROR

003614  022707  100000          5$:     CMP     #100000,PC      ;See if the program is high enough.
003620  101415                          BLOS    6$              ;
003622  062707  020000                  ADD     #20000,PC       ;If not, jump to the copy in the next 4K.
003626  062706  020000                  ADD     #20000,SP       ;Adjust the stack pointer to the current area.
003632  062716  020000                  ADD     #20000,(SP)     ;Adjust all the returns.
003636  062766  020000  000002          ADD     #20000,2(SP)    ;
003644  062766  020000  000004          ADD     #20000,4(SP)    ;
003652  000724                          BR      BUMP            ;Bump again.

003654                          6$:     RETURN
003654  000207                          RTS     PC
```

```
                        .SBTTL SPARTY - Initialize the parity CSR table and set up WRNGPA
                        ;*****************************************************************************
                        ;INPUTS:        [IE1]   Address of WRNGPA.
                        ;               [DI1] - Address of PTYTBL.
                        ;
                        ;OUTPUTS:       [DO1] - PTYTBL is set up with the addresses of all "live"
                        ;                       parity CSRs and -4 for all error counts.
                        ;               [DO2] - WRNGPA is set up with a "write wrong parity" command
                        ;                       for each "live" parity CSR.
                        ;
                        ;CALLING SEQUENCE:      CALL    SPARTY,IE1
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [*CL1]-ERROR
                        ;*****************************************************************************
003656  000240          SPARTY: NOP                             ;UFD 03/13/84
003660                          PRESERVE                        ;Preserve registers.
003660  004567  002166          JSR     R5,PREGO5
003664  012704  001000          MOV     #PTYTBL,R4              ;Get the address of PTYTBL.
003670  010403                  MOV     R4,R3                   ;
003672  012702  172100          MOV     #172100,R2              ;Set up the address of the parity CSRs.
003676  012701  000020          MOV     #16.,R1                 ;Set up to check for all CSRs.
003702                          PUSH    @#TIMOUT                ;Save the current time-out handler.
003702  013746  000004          MOV     @#TIMOUT,-(SP)
003706  004767  000066          JSR     PC,4$                   ;Get the address of the local time-out handler.
003712  012637  000004          MOV     (SP)+,@#TIMOUT          ;Set up the timeout vector.

003716  012712  000000  1$:     MOV     #0,(R2)                 ;Test for a CSR.
003722  001006                  BNE     2$                      ;

003724  012725  000005          MOV     #5,(R5)+                ;If the CSR exists, make entr' s in WRNGPA.
003730  010225                  MOV     R2,(R5)+                ;
003732  010224                  MOV     R2,(R4)+                ;Make entries in PTYTBL.
003734  012724  177774          MOV     #-4,(R4)+               ;

003740  062702  000002  2$:     ADD     #2,R2                   ;Bump to the next CSR address.
003744  005065  000002          CLR     2(R5)                   ;Ensure a zero word at the end of each table.
003750  005014                  CLR     (R4)                    ;
003752  077117                  SOB     R1,1$                   ;Test all 16 possible CSR addresses.

003754                          POP     @#TIMOUT                ;Restore the time-out vector.
003754  012637  000004          MOV     (SP)+,@#TIMOUT

003760  020403                  CMP     R4,R3                   ;See if any CSRs were alive.
003762  001005                  BNE     3$                      ;
003764  004767  003074          JSR     PC,TNOPTY               ;
003770                          CALL    ERROR,(SP)+             ;If not, signal an error.
003770  012605                  MOV     (SP)+,R5
003772  004767  001446          JSR     PC,ERROR

003776  000207          3$:     RETURN
003776                          RTS     PC

                        ;
                        ;
004000  004736          4$:     JSR     PC,@(SP)+               ;Local timeout handler locater.
```

SPARTY - INITTALIZE THE PARITY CSR TABLE AND SET UP WRNGPA

```
   1189
   1190 004002  042766  000004  000002          BIC     #4,2(SP)        ;Clear the Z bit in the CSR.
   1191 004010  000002                          RTI                     ;Return from trap.
```

CPARTY - CHECK FOR ANY PARITY ERRORS

```
                                        .SBTTL CPARTY - Check for any parity errors
                                        ;*****************************************************************************
                                        ;INPUTS:        [DI1]   The address of PTYTBL.
                                        ;
                                        ;OUTPUTS:       [*AD01] - Error message to the console.
                                        ;
                                        ;CALLING SEQUENCE:      CALL    CPARTY
                                        ;
                                        ;SUBORDINATE ROUTINES CALLED:   [*CL1]-PTYERR
                                        ;*****************************************************************************

004012  000240                  CPARTY: NOP                             ;UFD 03/13/84
004014                                  PRESERVE                        ;Preserve registers.
004014  004567  002032                  JSR     R5,PREG05

004020  012700  001000                  MOV     #PTYTBL,R0              ;Get the address of PTYTBL.

004024  005720                  1$:     TST     (R0)+                   ;Check for end of table.
004026  001407                          BEQ     2$                      ;

004030  005720                          TST     (R0)+                   ;Check the error count.
004032  001774                          BEQ     1$                      ;
004034                                  CALL    PTYERR,-4(R0)           ;If error, signal which CSR detected the error.
004034  016005  177774                  MOV     -4(R0),R5
004040  004767  000726                  JSR     PC,PTYERR
004044  000767                          BR      1$                      ;

004046                          2$:     RETURN
004046  000207                          RTS     PC
```

```
                        .SBTTL CLRMEM - Clear all selected memory
                        ;*****************************************************************************
                        ;INPUTS:        [IE1]   The low PAR setting to be used.
                        ;               [IE2] - The high PAR setting to be used.
                        ;               [II1] - Memory Management is set up and enabled (22 bits).
                        ;
                        ;OUTPUTS:       [AD01] - All selected memory is set to all zeroes.
                        ;
                        ;CALLING SEQUENCE:      CALL    CLRMEM,IE1,IE2
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [CL1]-CLEAR     [*CL2]-SEGERR
                        ;*****************************************************************************

004050  000240          CLRMEM: NOP                     ;UFD 03/13/84
004052                          PRESERVE                ;Preserve registers.
004052  004567  001774          JSR     R5,PREG05

004056  010501                  MOV     R5,R1           ;Move IE1 to a safe register.
004060  010402                  MOV     R4,R2           ;And IE2.

004062  012705  020000          MOV     #20000,R5       ;Set up PE1 for CLEAR.
004066  012704  010000          MOV     #4096.,R4       ;And PE2.

004072  010137  172342  1$:     MOV     R1,@#KPAR1      ;Set up the working PAR.
004076                          CALL    CLEAR           ;Clear the 4096 word segment.
004076  004767  000266          JSR     PC,CLEAR

004102  005700                  TST     R0              ;Check the returned test flag.
004104  001402                  BEQ     2$              ;
004106                          CALL    SEGERR          ;If error, signal an error in the segment.
004106  004767  001154          JSR     PC,SEGERR

004112  062701  000200  2$:     ADD     #200,R1         ;Bump the PAR setting.
004116  020102                  CMP     R1,R2           ;See if all selected memory is cleared.
004120  101764                  BLOS    1$              ;If not, do the next segment.

004122                          RETURN
004122  000207                  RTS     PC
```

```
                                .SBTTL TSTMEM - Test all selected memory
                                ;*************************************************************************
                                ;INPUTS:        [IE1]   The low PAR setting.
                                ;               [IE2] - The high PAR setting.
                                ;               [II1] - Memory Management is set up and enabled (22 bits).
                                ;
                                ;OUTPUTS:       [AD01] - All memory selected by IE1 and IE2 is tested.
                                ;
                                ;CALLING SEQUENCE:      CALL    TSTMEM,IE1,IE2
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [#CL1]-INVMEM
                                ;*************************************************************************

004124  000240                  TSTMEM: NOP                             ;UFD 03/13/84
004126                                  PRESERVE                        ;Preserve registers.
004126  004567  001720                  JSR     R5,PREG05

004132  012703  000001                  MOV     #1,R3                   ;Initialize PE3, the address increment bit.
004136  005002                          CLR     R2                      ;Initialize the pos/neg flag, PE4.
004140  012700  000014                  MOV     #12.,R0                 ;Initialize the bit shift count.

004144  006303                  1$:     ASL     R3                      ;Shift the address increment bit.
004146  012701  010000                  MOV     #4096.,R1               ;Inverting from zeroes to ones.
004152                                  CALL    INVMEM                  ;Invert all memory, positive address pattern.
004152  004767  000250                  JSR     PC,INVMEM

00.156  005102                          COM     R2                      ;Shift to a negative addressing pattern.
004160  005001                          CLR     R1                      ;Inverting from ones to zeroes.
004162                                  CALL    INVMEM                  ;Invert all memory, negative address pattern.
004162  004767  000240                  JSR     PC,INVMEM
004166  012701  010000                  MOV     #4096.,R1               ;Inverting from zeroes to ones.
004172                                  CALL    INVMEM                  ;Invert all memory, negative address pattern.
004172  004767  000230                  JSR     PC,INVMEM

004176  005102                          COM     R2                      ;Shift back to a positive addressing pattern.
004200  005001                          CLR     R1                      ;Inverting from ones to zeroes.
004202                                  CALL    INVMEM                  ;Invert all memory, positive address pattern.
004202  004767  000220                  JSR     PC,INVMEM

004206  077022                          SOB     R0,1$                   ;Test all 12 addressing patterns.

004210                                  RETURN
004210  000207                          RTS     PC
```

```
                                .SBTTL SETLOW   Relocate and remap for operation in the low 16K words
                                ;*****************************************************************************
                                ;INPUTS:        [DI1]   TKMAP2, the address of the PAR map for low 16K
                                ;                       operation.
                                ;
                                ;OUTPUTS:       [DO1] - RELFLG is set.
                                ;               [ADO1] - The program is relocated to the low 16K words of
                                ;                        memory and the KT is remapped for operation there.
                                ;
                                ;CALLING SEQUENCE:      CALL    SETLOW
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-RLCATE    [CL2]-SETABL
                                ;*****************************************************************************

004212  000240                 SETLOW: NOP                             ;UFD 03/13/84
004214                                  PRESERVE                        ;Preserve registers.
004214  004567  001632                  JSR     R5,PRESOS

004220  012737  000000  172342          MOV     #PAROK,@#KPAR1  ;Ensure PAR 1 is properly set.

004226                                  CALL    RLCATE                  ;Relocate the program in physical memory.
004226  004767  000326                  JSR     PC,RLCATE

004232  004767  002034                  JSR     PC,TKMAP2       ;Get the address of TKMAP2.
004236                                  CALL    SETABL,(SP)+    ;Pass it to SETABL.
004236  012605                          MOV     (SP)+,R5
004240  004767  000102                  JSR     PC,SETABL

004244  012767  000001  001644          MOV     #1,RELFLG       ;Set the relocation flag.

004252                                  RETURN
004252  000207                          RTS     PC
```

```
                        .SBTTL SETBAK - Relocate back to original physical memory and Reset
                        ;*****************************************************************************
                        ;INPUTS:        [DI1]   RELFLG.
                        ;               [DI2] - TKMAP4, the address of the PDR map for all APRs set to
                        ;                       Read/Write.
                        ;
                        ;OUTPUTS:       [OI1] - The processor is in a Reset state.
                        ;               [DO1] - RELFLG is cleared.
                        ;               [ADO1] - The program resides in the 16K to 20K physical
                        ;                        memory area.
                        ;
                        ;CALLING SEQUENCE:      CALL    SETBAK
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   [*CL1]-SETABL    [*CL2]-RLCATE
                        ;*****************************************************************************

004254  000240          SETBAK: NOP                             ;UFD 03/13/84
004256                          PRESERVE                        ;Preserve registers.
004256  004567  001570          JSR     R5,PREG05

004262  032767  077777  001626          BIT     #77777,RELFLG   ;See if a relocation is needed.
004270  001414                  BEQ     1$                      ;

004272  004767  002070          JSR     PC,TKMAP4               ;Get the address of TKMAP4.
004276                          CALL    SETABL,(SP)+            ;Pass it to SETABL.
004276  012605                  MOV     (SP)+,R5
004300  004767  000042          JSP     PC,SETABL

004304  012737  001000  172342  1$:     MOV     #PAR16K,@#KPAR1 ;Ensure that PAR 1 is properly set.

004312  005067  001600          CLR     RELFLG                  ;Reset RELFLG.

004316                          CALL    RLCATE                  ;Relocate back to original physical memory.
004316  004767  000236          JSR     PC,RLCATE

004322  000005                  RESET

004324  056706  001566          BIS     RELFLG,SP               ;Fix the stack pointer.

004330  005000                  CLR     R0                      ;Do a timer to allow the Reset to settle.
004332  000240          2$:     NOP                             ;
004334  077002                  SOB     R0,2$                   ;

004336  000240          3$:     NOP                             ;                       ;UFD 03/13/84
004340  077002                  SOB     R0,3$                   ;                       ;UFD 03/13/84
004342  000240                  NOP                             ;                       ;UFD 03/13/84

004344                          RETURN
004344  000207                  RTS     PC
```

```
                        .SBTTL SETABL - Set locations from a Data-Address table
                        ;*****************************************************************************
                        ;INPUTS:        [IE1]   The address of the table to be used.
                        ;
                        ;OUTPUTS:       [D01] - The locations addressed by the table are set up with
                        ;                       data from the table.
                        ;
                        ;CALLING SEQUENCE:      CALL    SETABL,IE1
                        ;
                        ;COMMENTS:      The table sequence is:
                        ;                               End   - )-      0       )
                        ;                                       )-      .       )
                        ;                                       )- Address Slot -)
                        ;                                       )- Data Slot    -)
                        ;                                       )-      .       -)
                        ;                                       )-      .       -)
                        ;                                       )-      .       -)
                        ;                                       )- Address Slot -)
                        ;                                       )  Data Slot    -)
                        ;                                       )- Address Slot -)
                        ;                               Start - )- Data Slot    -)
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   None.
                        ;*****************************************************************************

004346  000240          SETABL: NOP                     ;MOD 03/13/84
004350                          PRESERVE                ;Preserve registers.
004350  004[..  .1476           JSR     R5,PREG05

004354  012504          1$:     MOV     (R5)+,R4        ;Get the data.

004356  00[..5                  TST     (R5)            ;Check for end of table.
004360  001402                  BEQ     2$              ;

004362  01[..3                  MOV     R4,@(R5)+       ;If not the end, put the data in the location.
004364  00[..3                  BR      1$              ;Continue until end of table.

004366                  2$:     RETURN
004366  000207                  RTS     PC
```

```
                        .SBTTL CLEAR  - Clear a segment of memory
                        ;*****************************************************************************
                        ;INPUTS:        [IE1] The starting address of the segment to be cleared.
                        ;               [IE2] The segment size in words.
                        ;
                        ;OUTPUTS:       [DO1] The segment of memory is cleared.
                        ;               [OE1] zero if no errors are detected, non-zero if any errors
                        ;                       are detected.
                        ;
                        ;CALLING SEQUENCE:      CALL    CLEAR  IE1,IE2
                        ;
                        ;COMMENTS:      The selected memory segment is cycled back and forth between
                        ;               all ones and all zeros (to warm it up).  This is done with
                        ;               byte addressing.
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   None.
                        ;*****************************************************************************

004370  000240          CLEAR:  NOP                     ;UFD 03/13/84
004372                          PRESERVE                ;Preserve registers.
004372  004567  001454          JSR     R5,PREG05

004376  005000                  CLR     R0              ;Initialize a zero.

004400  110015          1$:     MOVB    R0,(R5)         ;Set the low byte to zero.
004402  105115                  COMB    (R5)            ;Complement it.
004404  105115                  COMB    (R5)            ;Set it back to zero.

004406  112515                  MOVB    (R5)+,(R5)      ;Clear the high byte.
004410  105115                  COMB    (R5)            ;Complement it.
004412  105115                  COMB    (R5)            ;Set it back to zero.

004414  112500                  MOVB    (R5)+,R0        ;Put it in R0 for transfer to the next byte.

004416  077410                  SOB     R4,1$           ;Do the complete segment.

004420                          RETURN  R0              ;Return the error flag [OE1].
004420  010066  000002          MOV     R0,ROSLOT(SP)
004424  000207                  RTS     PC
```

```
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:       [ADO1] - All selected memory is inverted and checked for
1452                                    ;                       errors.
1453                                    ;               [*ADO2]- 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.
     004430  004567  001416                    JSR     R5,PREG05
1462 004434  005702                            TST     R2              ;Check the pos/neg flag.
1463 004436  001010                            BNE     1$              ;
1464
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.
     004444  010305                            MOV     R3,R5
     004446  004767  000424                    JSR     PC,SPLADR
1468 004452  012703  000200                    MOV     #200,R3         ;
1469 004456  000411                            BR      3$              ;
1470 004460  010401                    1$:     MOV     R4,R1           ;If negative, set IE2 as the working setting.
1471 004462  010502                            MOV     R5,R2           ;Set IE1 as the terminal PAR setting.
1472 004464                                    CALL    SMIADR,R3       ;Set up a negative addressing pattern.
     004464  010305                            MOV     R3,R5
     004466  004767  000460                    JSR     PC,SMIADR
1473 004472  012703  177600                    MOV     #-200,R3        ;
1474 004476  000401                            BR      3$              ;
1475
1476 004500  060301                    2$:     ADD     R3,R1           ;Update the PAR setting.
1477
1478 004502  016604  000004            3$:     MOV     R1SLOT(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.
     004512  012605                            MOV     (SP)+,R5
     004514  004767  177626                    JSR     PC,SETABL
1481
1482 004520  010167  172342                    MOV     R1,@#KPAR1      ;Set up the working PAR.
1483 004524                                    CALL    INVERT          ;Invert a 4K word segment.
     004524  004767  000500                    JSR     PC,INVERT
1484
1485 004530  060400                            ADD     R4,R0           ;Check for errors.
1486 004532  001403                            BEQ     4$              ;
1487 004534                                    CALL    SEGERR          ;If error, signal an error in the segment.
     004534  004767  000526                    JSR     PC,SEGERR
```

INVMEM - INVERT ALL SELECTED MEMORY

```
1488
1489 004540  020100            4$:     CMP     R1,R2           ;Check done.
1490 004542  001356                     BNE     2$              ;
1491
1492 004544  004767  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  011605                     MOV     (SP)+,R5
     004552  004767  177570             JSR     PC,SETABL
1494 004556                              RETURN
     004556  000207                     RTS     PC
```

```
                                .SBTTL RLCATE - Relocate the program
                                ;*****************************************************************************
                                ;INPUTS:        None.
                                ;
                                ;OUTPUTS:       [D01] - A copy of the contents of memory from 100000 to 157776
                                ;                       is made in memory from 20000 to 77776
                                ;
                                ;CALLING SEQUENCE:      CALL    RLCATE
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [*CL1]-ERROR
                                ;*****************************************************************************
                                ;
004560  000240                 RLCATE: NOP                             ;UFD 03/13/84
004562                                  PRESERVE                        ;Preserve registers.
004562  004567  001264                  JSR     R5,PREG05
004566  012700  100000                  MOV     #100000,R0      ;Set the base address for the first checksum.
004572  012701  030000                  MOV     #12288.,R1      ;Set the word count.
004576  005005                          CLR     R5              ;Initialize the first checksum.
                                ;
004600  062005                  1$:     ADD     (R0)+,R5        ;Generate the first checksum.
004602  005505                          ADC     R5              ;Add the carry bit back in.
004604  077103                          SOB     R1,1$           ;
                                ;
004606  012700  100000                  MOV     #100000,R0      ;Set the base address for the move.
004612  012701  020000                  MOV     #20000,R1       ;
004616  012702  030000                  MOV     #12288.,R2      ;Set word count.
                                ;
004622  012021                  2$:     MOV     (R0)+,(R1)+     ;Move the block.
004624  077202                          SOB     R2,2$           ;
                                ;
004626  012700  020000                  MOV     #20000,R0       ;Set the base address for the second checksum.
004632  012701  030000                  MOV     #12288.,R1      ;Set word count.
004636  005004                          CLR     R4              ;Initialize the second checksum.
                                ;
004640  062004                  3$:     ADD     (R0)+,R4        ;Generate the second checksum.
004642  005504                          ADC     R4              ;Add the carry bit back in.
004644  077103                          SOB     R1,3$           ;
                                ;
004646  020405                          CMP     R4,R5           ;Check the checksums.
004650  001405                          BEQ     4$              ;
004652  004767  002166                  JSR     PC,TLOW32       ;
004656                                  CALL    ERROR,(SP)+     ;If not a match, signal a bad relocation.
004656  012605                          MOV     (SP)+,R5
004660  004767  000560                  JSR     PC,ERROR
                                ;
004664                          4$:     RETURN
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.
     004670  004567  001156                    JSR     R5,PREG05
1555
1556 004674  004767  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.
     004700  012605                            MOV     (SP)+,R5
     004702  004767  000536                    JSR     PC,ERROR
1558
1559 004706                                     CALL    WRITE,@ETVCTR           ;Display "this is the trapping vector".
     004706  012705  001504                    MOV     @ETVCTR,R5
     004712  004767  000566                    JSR     PC,WRITE
1560
1561 004716                                     CALL    WRIUSN,R5SLOT(SP),#'O,#8.  ;Display the trapping vector.
     004716  016605  000014                    MOV     R5SLOT(SP),R5
     004722  012704  000060                    MOV     #'O,R4
     004726  012703  000010                    MOV     #8.,R3
     004732  004767  000702                    JSR     PC,  IUSN
1562
1563 004736  005767  001156                    TST     [illegible] IN          ;ARE WE IN UFD F.S. MODE?       ;UFD 03/08/84
1564 004742  001405                            BEQ                             ;BR IF NOT                      ;UFD 03/08/84
1565 004744  012737  000032  001104            MOV     #32,@#UFDCHR            ;MAKE LIKE ^Z TYPED             ;UFD 03/08/84
1566 004752                                     CALL    CTRLX                   ;LET'S GET OUT                  ;UFD 03/08/84
     004752  004767  175310                    JSR     PC,CTRLX
1567 004756                                                                                                     ;UFD 03/08/84
1568 004756  004767  002176                    JSR     PC,TGCTU                ;
1569 004762                                     CALL    WRITPG,(SP)+            ;Display "continuing with the test"
     004762  016705                            MOV     (SP)+,R5
     004764  004767  000544                    JSR     PC,WRITPG
1570
1571 004770                                     RETURN
     004770  000207                            RTS     PC
```

PTYERR - SIGNAL ' PARITY DETECT ERROR

```
1572                                    .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.
     004774  004567  001052                    JSR     R5,PREGO5
1588
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.
     005004  012605                            MOV     (SP)+,R5
     005006  004767  000432                    JSR     PC,ERROR
1591
1592 005012                                     CALL    WRITE,@EPACSR           ;Display "this is the parity CSR".
     005012  012705  002010                    MOV     @EPACSR,R5
     005016  004767  000562                    JSR     PC,WRITE
1593
1594 005022                                     CALL    WRIUSN,R5SLOT(SP),#'0,#8.  ;Display the CSR address.
     005022  016605  000014                    MOV     R5SLOT(SP),R5
     005026  012704  000060                    MOV     #'0,R4
     005032  012703  000010                    MOV     #8.,R3
     005036  004767  000576                    JSR     PC,WRIUSN
1595
1596 005042  005767  001052                    TST     UFDCHN                  ;ARE WE IN UFD F.S. MODE?       ;UFD 03/08/84
1597 005046  001405                            BEQ     1$                      ;BR IF NOT                      ;UFD 03/08/84
1598 005050  012737  000032  001104            MOV     #32,@#UFDCHR            ;MAKE LIKE ^Z TYPED             ;UFD 03/08/84
1599 005056                                     CALL    CTRLX                   ;LET'S GET OUT                  ;UFD 03/08/84
     005056  004767  175204                    JSR     PC,CTRLX
1600 005062                                1$:                                                                  ;UFD 03/08/84
1601 005062  004767  002072                    JSR     PC,TGCTU                ;
1602 005066                                     CALL    WRITPG,(SP)+            ;Display "continuing with the test"
     005066  012605                            MOV     (SP)+,R5
     005070  004767  000440                    JSR     PC,WRITPG
1603
1604 005074                                     RETURN
     005074  000207                            RTS     PC
```

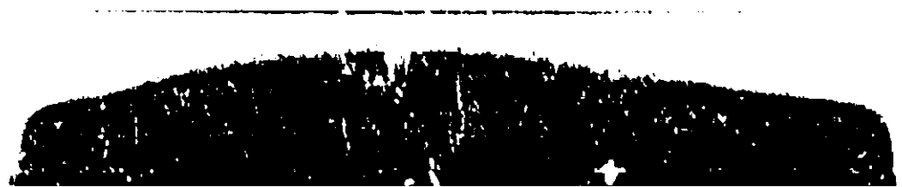
SPLADR - SET THE SEGMENT ADDRESS TABLE WITH A POSITIVE ADDRESSI

```
                                .SBTTL SPLADR - Set the Segment Address Table with a positive addressing pattern
                                ;*****************************************************************************
                                ;INPUTS:        [IE1]   The address bit to be used in developing
                                ;                       the address pattern.
                                ;
                                ;OUTPUTS:       [DO1] - The Segment Address Table is set up with a positive
                                ;                       addressing pattern.
                                ;
                                ;CALLING SEQUENCE:      CALL    SPLADR
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   None
                                ;*****************************************************************************

005076  000240                  SPLADR: NOP                             ;CFD 02.03.04
005100                                  PRESERVE                        ;Preserve registers.
005100  004567  000746                  JSR     R5,PRESV3
                                
005104  012704  120000                  MOV     #120000,R4              ;Set the starting address of the segment address
                                                                        ;table.
005110  012703  020000                  MOV     #20000,R3               ;Set the base address to be used (PAR 1 range).
005114  012702  010000                  MOV     #4096.,R2               ;Set the word count.
005120  012701  160000                  MOV     #160000,R1              ;Establish a mask for a 4K word range.
005124  005000                          CLR     R0                      ;Initialize the address word.
                                
005126  010014          1$:     MOV     R0,(R4)                 ;Place the address in the Segment Address
                                                                ;table.
005130  060324                  ADD     R3,(R4)+                ;
                                
005132  060500                  ADD     R5,R0                   ;Update the address.
005134  030100                  BIT     R1,R0                   ;Check for an overflow.
005136  001403                  BEQ     2$                      ;
                                
005140  062700  000002                  ADD     #2,R0           ;If overflow, add in the overflow increment.
005144  040100                  BIC     R1,R0                   ;Mask out the overflow bit.
                                
005146  077211          2$:     SOB     R2,1$                   ;Continue until the table is full.
                                
005150                          RETURN
005150  000207                  RTS     PC
```

```
                        .SBTTL SMIADR - Set the Segment Address Table with a negative addressing pattern
                        ;******************************************************************************
                        ;INPUTS:        [IE1]   The address bit to be used in developing
                        ;                       the address pattern.
                        ;
                        ;OUTPUTS:       [DO1] - The Segment Address Table is set up with a negative
                        ;                       addressing pattern.
                        ;
                        ;CALLING SEQUENCE:      CALL    SMIADR,IE1
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   None.
                        ;******************************************************************************

005152  000240          SMIADR: NOP                     ;UFD 03/13/84
005154                          PRESERVE                ;Preserve registers.
005154  004567  000672          JSR     R5,PREG05

005160  012704  120000          MOV     #120000,R4      ;Set up the address of the segment address
                                                        ;table.
005164  012703  020000          MOV     #20000,R3       ;Set the base address to be used (PAR 1 range).
005170  012702  010000          MOV     #4096.,R2       ;Set word count.
005174  012701  160000          MOV     #160000,R1      ;Establish a mask for a 4K word range.
005200  012700  017776          MOV     #17776,R0       ;Initialize the address word.

005204  010014          1$:     MOV     R0,(R4)         ;Place the address in the Segment Address
                                                        ;table.
005206  060324                  ADD     R3,(R4)+        ;

005210  160500                  SUB     R5,R0           ;Update the address.
005212  030100                  BIT     R1,R0           ;Check for an underflow.
005214  001403                  BEQ     2$              ;

005216  162700  000002          SUB     #2,R0           ;If underflow, subtract underflow increment.
005222  040100                  BIC     R1,R0           ;Mask out the underflow bits.

005224  077211          2$:     SOB     R2,1$           ;Continue until the table is full.

005226                          RETURN
005226  000207                  RTS     PC
```

```
                                .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.
                                ;******************************************************************************

005230  000240                  INVERT: NOP                     ;UFD 03/13/84
005232                                  PRESERVE                ;Preserve registers.
005232  004567  000614                  JSR     R5,PREG05

005236  012705  120000                  MOV     #120000,R5      ;Starting address of the segment address table.
005242  005000                          CLR     R0              ;Initialize the test flag.

005244  012703  010000                  MOV     #4096.,R3       ;Set the word count.

005250  012504          1$:     MOV     (R5)+,R4        ;Get the address to be tested.
005252  005114                  COM     (R4)            ;Invert it.
005254  061400                  ADD     (R4),R0         ;Update the test flag.
005256  077304                  SOB     R3,1$           ;Do the complete segment.

005260                          RETURN  R0              ;Return the test flag [OE1].
005260  010066  000002          MOV     R0,ROSLOT(SP)
005264  000207                  RTS     PC
```

```
                                .SBTTL SEGERR - Signal an error in a segment of memory
                                ;*****************************************************************************
                                ;INPUTS:        [DI1]   The contents of PAR 1, the current memory segment
                                ;                       under test.
                                ;
                                ;OUTPUTS.       [AD01] - Memory segment error messages are displayed on the
                                ;                        console.
                                ;
                                ;CALLING SEQUENCE:      CALL    SEGERR
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-ERROR     [CL2]-WRITE     [CL3]-WRIUSN
                                ;                               [CL4]-WRITPG    [CL5]-SETBAK    [CL6]-MAIN1
                                ;*****************************************************************************

005266  013700  172342          SEGERR: MOV     @#KPAR1,R0      ;Get the current contents of PAR 1.

005272  004767  001712                  JSR     PC,TSEGER       ;Get the address of the message table.
005276                                  CALL    ERROR,(SP)+     ;Signal the error and wait for Control E.
005276  012605                          MOV     (SP)+,R5
005300  004767  000140                  JSR     PC,ERROR

005304                                  CALL    WRITE,#ESEGAD   ;Display the "this is the bad segment" message.
005304  012705  002143                  MOV     #ESEGAD,R5
005310  004767  000270                  JSR     PC,WRITE

005314                                  CALL    WRIUSN,R0,#'0,#8.  ;Display the bad segment address.
005314  010005                          MOV     R0,R5
005316  012704  000060                  MOV     #'0,R4
005322  012703  000010                  MOV     #8.,R3
005326  004767  000306                  JSR     PC,WRIUSN

005332  005767  000562                  TST     UFDCHN          ;ARE WE IN UFD F.S. MODE?       ;UFD 03/08/84
005336  001405                          BEQ     1$              ;BR IF NOT                      ;UFD 03/08/84
005340  012737  000032  001104          MOV     #32,@#UFDCHR    ;MAKE LIKE ^Z TYPED             ;UFD 03/08/84
005346                                  CALL    CTRLX           ;LET'S GET OUT                  ;UFD 03/08/84
005346  004767  174714                  JSR     PC,CTRLX
005352                          1$:                                                             ;UFD 03/08/84
005352  004767  001550                  JSR     PC,TRSTRT       ;Get the address of the "restarting" message
                                                                ;address table.
005356                                  CALL    WRITPG,(SP)+    ;Display the message.
005356  012605                          MOV     (SP)+,R5
005360  004767  000150                  JSR     PC,WRITPG

005364                                  CALL    SETBAK          ;Ensure the program is ready for restart.
005364  004767  176664                  JSR     PC,SETBAK

005370                                  CALL    MAIN1           ;Go restart the program.
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                                    CALL    SETBAK          ;RESTORE PGM TO ORIG LOC.       ;UFD 03/09/84
     005402  004767  176646                    JSR     PC,SETBAK
1765
1766 005406  016737  000512  000030            MOV     SAV30,@#30      ;Restore EMT location 30
1767 005414  016737  000506  000032            MOV     SAV32,@#32      ;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 a -1 into location  drserr 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:       [ADO1] - The standard customer error message is displayed on
1782                                    ;                        the console.
1783                                    ;               [*ADO2]- 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                        ;Preserve registers.
     005446  004567  000400                     JSR     R5,PREG05
1794
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.
     005462  012605                             MOV     (SP)+,R5
     005464  004767  000044                     JSR     PC,WRITPG
1798
1799 005470                            1$:      CALL    GETCHR                  ;Get an input character.
     005470  004767  174410                     JSR     PC,GETCHR
1800 005474  020027  000030                     CMP     R0,#30                  ;Check for Control X.
1801 005500  001002                             BNE     2$                      ;
1802 005502                                     CALL    CTRLX                   ;If Control X, exit via CTRLX.
     005502  004767  174560                     JSR     PC,CTRLX
1803
1804 005506  005767  000406            2$:      TST     UFDCHN                  ;ARE WE IN UFD F.S. MODE?       ;UFD 03/08/84
1805 005512  001003                             BNE     3$                      ;BR IF WE ARE                   ;UFD 03/08/84
1806 005514  020027  000005                     CMP     R0,#5                   ;Check for Control E.
1807 005520  001363                             BNE     1$                      ;If not Control E, wait in loop.
1808
1809 005522                            3$:      CALL    WRITPG,R5SLOT(SP)       ;Display the expanded error information.
     005522  016605  000014                     MOV     R5SLOT(SP),R5
     005526  004767  000002                     JSR     PC,WRITPG
1810
1811 005532                                     RETURN
     005532  000207                             RTS     PC
```

```
                        .SBTTL WRITPG  (WRITE PACKAGE) Write several ASCIZ strings with returns
                        ;*****************************************************************************
                        ;INPUTS:        [IE1]   R5 contains a table address.  That table contains
                        ;               the offsets (into the text table) of the strings to be
                        ;               printed in order.  The table must be terminated by a zero.
                        ;
                        ;OUTPUTS:       [AD01] The strings are displayed on the console.
                        ;
                        ;CALLING SEQUENCE:      CALL    WRITPG,IE1
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   WRITLN.
                        ;*****************************************************************************

005534                  WRITPG: PUSH    R5,R4           ;Save registers.
005534  010546                  MOV     R5,-(SP)
005536  010446                  MOV     R4,-(SP)

005540  010504                  MOV     R5,R4           ;Put the table address in R4.

005542  012405          1$:     MOV     (R4)+,R5        ;Put the string offset in R5 for WRITLN.
005544  001403                  BEQ     2$              ;If terminator found, get out.

005546                          CALL    WRITLN          ;Print the string as a line.
005546  004767  000010          JSR     PC,WRITLN
005552  000773                  BR      1$              ;And get the next string.

005554                  2$:     POP     R4,R5           ;Restore registers.
005554  012604                  MOV     (SP)+,R4
005556  012605                  MOV     (SP)+,R5
005560                          RETURN
005560  000207                  RTS     PC
```

```
                                .SBTTL WRITE  - (WRITE PACKAGE) Write an ASCIZ string to the console
                                ;******************************************************************
                                ;INPUTS:        [IE1]   R5 contains the offset (into the text area) of the
                                ;                      ASCIZ string.
                                ;
                                ;OUTPUTS:       [AD01] - The ASCIZ string is written to the console.
                                ;
                                ;CALLING SEQUENCE       CALL    WRITE,IE1
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-PRTCHR.
                                ;******************************************************************

005604                          WRITE:  PUSH    R5,R4           ;Save registers.
005604  010546                         MOV     R5,-(SP)
005606  010446                         MOV     R4,-(SP)

005610  004767  001512                 JSR     PC,LOCTXT       ;Get the text area address.
005614  012604                         MOV     (SP)+,R4        ;Put it in R4.
005616  060504                         ADD     R5,R4           ;Add in the offset.

005620  112405                  1$:    MOVB    (R4)+,R5        ;Put a byte in R5 for PRTCHR.
005622  001403                         BEQ     2$              ;If end of string, get out.

005624                                 CALL    PRTCHR          ;If not, print the character.
005624  004767  174310                 JSR     PC,PRTCHR
005630  000773                         BR      1$              ;And get the next byte.

005632                          2$:    POP     R4,R5           ;Restore registers.
005632  012604                         MOV     (SP)+,R4
005634  012605                         MOV     (SP)+,R5
005636                                 RETURN
005636  000207                         RTS     PC
```

```
                                .SBTTL WRIUSN - (WRITE PACKAGE) Write an unsigned number to the console
                                ;*****************************************************************************
                                ;INPUTS:        [IE1]   R5 contains the number in binary.
                                ;               [IE2] - R4 contains 60 if leading zeros are to be printed,
                                ;                       40 if leading spaces are to be printed, zero
                                ;                       if neither.
                                ;                       WARNING - If IE1 and IE2 are both zero, nothing will
                                ;                             be printed.
                                ;               [IE3] - R3 contains 10 (base 10) if the number is to be
                                ;                       converted to Decimal, 8 if octal.  Default is binary.
                                ;
                                ;OUTPUTS:       [AD01] The number is displayed on the console terminal.
                                ;
                                ;CALLING SEQUENCE:      CALL    WRIUSN,IE1,IE2,IE3
                                ;
                                ;COMMENTS:      This is a two page module.
                                ;
                                ;SUBORDINATE ROUTINES CALLED:   [CL1]-PRTCHR.
                                ;*****************************************************************************
005640  000240                  WRIUSN: NOP                             ;UFD 03/13/84
005642                                  PRESERVE                        ;Preserve registers.
005642  004567  000204                  JSR     R5,PREG05
005646  010500                          MOV     R5,R0                   ;Put the number in R0.
005650  005046                          CLR     -(SP)                   ;Create a slot for the character count.
005652  004767  000044                  JSR     PC,3$                   ;Set up for the local co-routine calls.

005656  022703  000010                  CMP     #8.,R3                  ;Check for octal printing.
005662  001434                          BEQ     5$                      ;Go handle octal.
005664  022703  000012                  CMP     #10.,R3                 ;Check for decimal printing.
005670  001434                          BEQ     6$                      ;Go handle decimal.

005672  012701  000020                  MOV     #16.,R1                 ;Set loop count to print 16 bits in binary.
005676  005005                  1$:     CLR     R5                      ;
005700  006100                          ROL     R0                      ;Put a bit in R5.
005702  006105                          ROL     R5                      ;
005704  004736                          JSR     PC,@(SP)+               ;The co-routine call prints the bit.
005706  077105                          SOB     R1,1$                   ;Print 16 bits.

005710                          2$:     POP     R0                      ;Purge the stack of the co-routine call.
005710  012600                          MOV     (SP)+,R0
005712                                  POP     R0                      ;Get the character count in R0.
005712  012600                          MOV     (SP)+,R0
005714                                  RETURN  R0                      ;Return the character count.
005714  010066  000002                  MOV     R0,ROSLOT(SP)
005720  000207                          RTS     PC

                                        ;+++
                                        ;This co-routine converts to ASCII and prints
                                        ;---

005722  004736                  3$:     JSR     PC,@(SP)+               ;Return from the co-routine.

005724  005705                          TST     R5                      ;Check the number for zero.
005726  001402                          BEQ     4$                      ;
005730  012704  000060                  MOV     #60,R4                  ;If not zero, set to print any later zeros.

005734  005704                  4$:     TST     R4                      ;Check the print flag.
```

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

```
1940 005736  001771                    BEQ     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                                        ;The following code converts to octal or decimal and prints.
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                       CLR     R5              ;Initialize the digit to be printed.
1966 006002  020100                  10$:  CMP     R1,R0           ;Compare the base with the remaining number.
1967 006004  101372                       BHI     8$              ;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

```
                        .SBTTL PREG05 - Preserve Registers R0 through R5 for subroutine calls
                        ;******************************************************************************
                        ;INPUTS:        The return address from the caller's caller must be the last
                        ;               entry on the stack.
                        ;
                        ;OUTPUTS:       Registers R0 through R5 are saved on the stack.  When
                        ;               the return is made from the calling subroutine, they are
                        ;               automatically restored.
                        ;
                        ;CALLING SEQUENCE:      PRESERVE
                        ;
                        ;COMMENTS:      This routine is re-entrant.
                        ;
                        ;               Parameters may be passed out of a subroutine by using the
                        ;               "RETURN" macro.
                        ;
                        ;SUBORDINATE ROUTINES CALLED:   None.
                        ;******************************************************************************

006052                  PREG05:                 ;R5 has been loaded on the stack by the call.
                                                ;R5 now contains the return to the calling routine.

006052                          PUSH    R4,R3,R2,R1,R0,R5
006052  010446                  MOV     R4,-(SP)
006054  010346                  MOV     R3,-(SP)
006056  010246                  MOV     R2,-(SP)
006060  010146                  MOV     R1,-(SP)
006062  010046                  MOV     R0,-(SP)
006064  010546                  MOV     R5,-(SP)
006066  016605  000014          MOV     R5SLOT(SP),R5   ;Restore R5 to its original value.

006072  004736                  JSR     PC,@(SP)+       ;Call the calling routine.

                        ;+++
                        ;The following code is executed when the calling routine
                        ;does a "RETURN".
                        ;---

006074                  POP     R0,R1,R2,R3,R4,R5
006074  012600          MOV     (SP)+,R0
006076  012601          MOV     (SP)+,R1
006100  012602          MOV     (SP)+,R2
006102  012603          MOV     (SP)+,R3
006104  012604          MOV     (SP)+,R4
006106  012605          MOV     (SP)+,R5

006110                  RETURN                  ;Return to the caller's caller.
006110  000207          RTS     PC
```

GLOBALS

```
                        .SBTTL  GLOBALS


006112  000000          SWADD:  .WORD   0               ;Will contain the address of SWSTAC in XXDP+.

006114  000000          XXRTN:  .WORD   0               ;Will contain the XXDP+ return address/Chain
                                                        ;Mode flag.

006116  000000          RELFLG: .WORD   0               ;Relocation flag.

006120  000000          UFDCHN: .WORD   0               ;UFD CHAIN MODE flag
006122  000000          UFDQUI: .WORD   0               ;UFD QUIET MODE FLAG            ;UFD 02/08/84

006124  000000          SAV30:  .WORD   0               ;Will contain contents of EMT vector
006126  000000          SAV32:  .WORD   0
006130  000000          SAV52:  .WORD   0               ;SAVE FOR LOC 52        ;UFD 02/08/84
006132  000000          SAVSP:  .WORD   0               ;SAVE FOR SP            ;UFD 02/08/84
006134  000000          SAVCAC: .WORD   0               ;SAVE FOR CACHE         ;UFD 04/03/84
006136  000000          ORIONF: .WORD   0               ;ORION CP FLAG          ;UFD 04/03/84

                        ;+++
                        ;These are used by the Write Package utility modules.
                        ;---

006140  177560          KBCSR:  .WORD   DEFCON          ;Keyboard CSR.
006142  177562          KBBUF:  .WORD   DEFCON+2        ;Keyboard buffer.
006144  177564          DSPCSR: .WORD   DEFCON+4        ;Display CSR.
006146  177566          DSPBUF: .WORD   DEFCON+6        ;Display buffer.
```

```
                                .SBTTL  TKMAP1 - (Data section) - Initial memory management and PSW map

006150  004736          TKMAP1: JSR     PC,@(SP)+       ;Table locater.

006152  004406                  .WORD   RW256           ;256 words, Read/Write.
006154  172300                  .WORD   KPDR0           ;Address of PDR 0.
006156  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006160  172302                  .WORD   KPDR1           ;Address of PDR 1.
006162  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006164  172304                  .WORD   KPDR2           ;Address of PDR 2.
006166  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006170  172306                  .WORD   KPDR3           ;Address of PDR 3.
006172  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006174  172310                  .WORD   KPDR4           ;Address of PDR 4.
006176  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006200  172312                  .WORD   KPDR5           ;Address of PDR 5.
006202  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006204  172314                  .WORD   KPDR6           ;Address of PDR 6.
006206  077406                  .WORD   RW4096          ;4096 words, Read/Write.
006210  172316                  .WORD   KPDR7           ;Address of PDR 7.

005212  001000                  .WORD   PAR16K          ;Base address at 16K.
006214  172340                  .WORD   KPAR0           ;Address of PAR 0.
006216  000000                  .WORD   PAR0K           ;Base address at 0.
006220  172342                  .WORD   KPAR1           ;Address of PAR 1.
006222  000200                  .WORD   PAR4K           ;Base address at 4K.
006224  172344                  .WORD   KPAR2           ;Address of PAR 2.
006226  000400                  .WORD   PAR8K           ;Base address at 8K.
006230  172346                  .WORD   KPAR3           ;Address of PAR 3.
006232  001000                  .WORD   PAR16K          ;Base address at 16K.
006234  172350                  .WORD   KPAR4           ;Address of PAR 4.
006236  001200                  .WORD   PAR20K          ;Base address at 20K.
006240  172352                  .WORD   KPAR5           ;Address of PAR 5.
006242  001400                  .WORD   PAR24K          ;Base address at 24K.
006244  172354                  .WORD   KPAR6           ;Address of PAR 6.
006246  177600                  .WORD   PARIO           ;Base address at I/O page.
006250  172356                  .WORD   KPAR7           ;Address of PAR 7.

006252  000020                  .WORD   MMU22A          ;Enable 22 bit mapping.
006254  172516                  .WORD   MMUSR3          ;Address of MMU SR3.
006256  000001                  .WORD   MMUENA          ;Enable memory management.
006260  177572                  .WORD   MMUSR0          ;Address of MMU SR0.

006262  000000                  .WORD   0               ;Processor priority zero, Kernal mode.
006264  177776                  .WORD   PSW             ;Address of PSW.

006266  000000                  .WORD   0               ;
006270  000000                  .WORD   0               ;Table terminator.
```

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


006272  004736        TKMAP2: JSR     PC,@(SP)+       ;Table locater.

006274  000000                .WORD   PAR0K           ;Base address at 0.
006276  172340                .WORD   KPAR0           ;Address of PAR 0.
006300  001000                .WORD   PAR16K          ;Base address at 16K.
006302  172342                .WORD   KPAR1           ;Address of PAR 1.
006304  001200                .WORD   PAR20K          ;Base address at 20K.
006306  172344                .WORD   KPAR2           ;Address of PAR 2.
006310  001400                .WORD   PAR24K          ;Base address at 24K.
006312  172346                .WORD   KPAR3           ;Address of PAR 3.
006314  000000                .WORD   PAR0K           ;Base address at 0.
006316  172350                .WORD   KPAR4           ;Address of PAR 4.
006320  000200                .WORD   PAR4K           ;Base address at 4K.
006322  172352                .WORD   KPAR5           ;Address of PAR 5.
006324  000400                .WORD   PAR8K           ;Base address at 8K.
006326  172354                .WORD   KPAR6           ;Address of PAR 6.
006330  000000                .WORD   0               ;
006332  000000                .WORD   0               ;Table terminator.
```

```
                                .SBTTL  TKMAP3 - (Data section) Memory Management Map for testing

TKMAP3: JSR     PC,@(SP)+       ;Table locate.

        .WORD   NR4096          ;Non-Resident.
        .WORD   KPDR2           ;Address of PDR 2.
        .WORD   NR4096          ;Non-Resident.
        .WORD   KPDR3           ;Address of PDR 3.
        .WORD   R04096          ;Read only.
        .WORD   KPDR4           ;Address of PDR 4.
        .WORD   R04096          ;Read only.
        .WORD   KPDR5           ;Address of PDR 5.
        .WORD   NR4096          ;Non-Resident.
        .WORD   KPDR6           ;Address of PDR 6.
        .WORD   0               ;
        .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  000736                     TKMAP4: JSR     PC,@(SP)+       ;Table locater.
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

```
                                .SBTTL  VCTORS - (Data area) Vectors table for UNXINT
                        
                        
                        
006420  004736          VCTORS: JSR     PC,@(SP)+       ;Table locater.
                        
006422  004767  173312  VCT000: JSR     PC,UNXINT
006426  004767  173306  VCT004: JSR     PC,UNXINT
006432  004767  173302  VCT010: JSR     PC,UNXINT
006436  004767  173276  VCT014: JSR     PC,UNXINT
006442  004767  173272  VCT020: JSR     PC,UNXINT
006446  004767  173266  VCT024: JSR     PC,UNXINT
006452  004767  173262  VCT030: JSR     PC,UNXINT
006456  004767  173256  VCT034: JSR     PC,UNXINT
                        
006462  004767  173252  VCT040: JSR     PC,UNXINT
006466  004767  173246  VCT044: JSR     PC,UNXINT
006472  004767  173242  VCT050: JSR     PC,UNXINT
006476  004767  173236  VCT054: JSR     PC,UNXINT
006502  004767  173232  VCT060: JSR     PC,UNXINT
006506  004767  173226  VCT064: JSR     PC,UNXINT
006512  004767  173222  VCT070: JSR     PC,UNXINT
006516  004767  173216  VCT074: JSR     PC,UNXINT
                        
006522  004767  173212  VCT100: JSR     PC,UNXINT
006526  004767  173206  VCT104: JSR     PC,UNXINT
006532  004767  173202  VCT110: JSR     PC,UNXINT
006536  004767  173176  VCT114: JSR     PC,UNXINT
006542  004767  173172  VCT120: JSR     PC,UNXINT
006546  004767  173166  VCT124: JSR     PC,UNXINT
006552  004767  173162  VCT130: JSR     PC,UNXINT
006556  004767  173156  VCT134: JSR     PC,UNXINT
                        
006562  004767  173152  VCT140: JSR     PC,UNXINT
006566  004767  173146  VCT144: JSR     PC,UNXINT
006572  004767  173142  VCT150: JSR     PC,UNXINT
006576  004767  173136  VCT154: JSR     PC,UNXINT
006602  004767  173132  VCT160: JSR     PC,UNXINT
006606  004767  173126  VCT164: JSR     PC,UNXINT
006612  004767  173122  VCT170: JSR     PC,UNXINT
006616  004767  173116  VCT174: JSR     PC,UNXINT
```

D6

SEQ 0068

VCTORS - (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)

```
                                .SBTTL  MESSAGE TABLES (Data section)

007022  004736          TKTERR: JSR     PC,@(SP)+       ;Table locater.

007024  000103                  .WORD   MEXINF          ;Expanded error message lines.
007026  000177                  .WORD   EKTER1          ;
007030  000310                  .WORD   EKTER2          ;Signal a Memory management abort.
007032  000422                  .WORD   EKTER3          ;
007034  000511                  .WORD   MCTUT1          ;
007036  000547                  .WORD   MCTUT2          ;
007040  000001                  .WORD   NEWLIN          ;
007042  000000                  .WORD   0               ;

007044  004736          TLOW32: JSR     PC,@(SP)+       ;Table locater.

007046  000103                  .WORD   MEXINF          ;Expanded error message lines.
007050  000614                  .WORD   EL32M1          ;
007052  000725                  .WORD   EL32M2          ;Signal an error in the low 32K words of
007054  000511                  .WORD   MCTUT1          ;memory.
007056  000547                  .WORD   MCTUT2          ;
007060  000001                  .WORD   NEWLIN          ;
007062  000000                  .WORD   0               ;

007064  004736          TNOPTY: JSR     PC,@(SP)+       ;Table locater.

007066  000103                  .WORD   MEXINF          ;Expanded error message lines.
007070  001026                  .WORD   ENOPT1          ;
007072  000511                  .WORD   MCTUT1          ;Signal a no parity detect situation.
007074  000001                  .WORD   NEWLIN          ;
007076  000000                  .WORD   0               ;

007100  004736          TCUSTM: JSR     PC,@(SP)+       ;Table locater.

007102  000001                  .WORD   NEWLIN          ;
007104  000001                  .WORD   NEWLIN          ;
007106  000001                  .WORD   NEWLIN          ;Standard customer error message.
007110  001117                  .WORD   ECUST1          ;
007112  000001                  .WORD   NEWLIN          ;
007114  001221                  .WORD   ECUST2          ;
007116  000001                  .WORD   NEWLIN          ;
007120  001261                  .WORD   ECUST3          ;
007122  001367                  .WORD   ECUST4          ;
007124  000000                  .WORD   0               ;
```

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
UFDCHR	001104	#10-385	*11-445	*15-538	19-749	*19-756	*19-761	19-762	19-768	*38-1565
		*39-1598	*43-1742							
UFDQUI	006122	*11-434	15-623	17-685	44-1761	#52-2038				
UNXERR	004666	14-598	#38-1553							
UNXINT	001740	#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	13-565	#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 CREF V01

SYMBOL	VALUE	REFERENCES								
VCT234	006656	#58-2215								
VCT240	006662	#58-2217								
VCT244	006666	#58-2218								
VCT250	006672	#58-2219								
VCT254	006676	#58-2220								
VCT260	006702	#58-2221								
VCT264	006706	#58-2222								
VCT270	006712	#58-2223								
VCT274	006716	#58-2224								
VCT300	006722	#58-2226								
VCT304	006726	#58-2227								
VCT310	006732	#58-2228								
VCT314	006736	#58-2229								
VCT320	006742	#58-2230								
VCT324	006746	#58-2231								
VCT330	006752	#58-2232								
VCT334	006756	#58-2233								
VCT340	006762	#58-2235								
VCT344	006766	#58-2236								
VCT350	006772	#58-2237								
VCT354	006776	#58-2238								
VCT360	007002	#58-2239								
VCT364	007006	#58-2240								
VCT370	007012	#58-2241								
VCT374	007016	#58-2242								
WRITE	005604	11-481	11-485	11-511	22-910	22-923	22-931	38 1559	39-1592	43-1736
		47-1854	47-1856	#48-1873						
WRITLN	005562	46-1833	#47-1852							
WRITPG	005534	38-1569	39-1602	43-1747	45-1797	45-1809	#46-1826			
WRIUSN	005640	22-908	22-918	22-926	38-1561	39 1594	43 1738	#49-1907		
WRNGPA	007220	23-964	#61-2331							
XOFF	= 000023	#5-232	17-696							
XON	= 000021	#5-231	17-701							
XXRTN	006114	*11-418	11-494	11-503	*20-815	44-1772	#52-2032			

MACRO CROSS REFERENCE CREF V01

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	22-918	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	25-998	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	06-263	12-527	13-556	17-689	23-968	28-1161	46-1826	47-1852	48-1873	51-2013
RETURN	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							

CVMSBB0 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