KDB50-1.MCR

KDB50_1.MCR;1

KDB50-1.MCR

LSCS FORM=FLAG

TIME QUEUED			TIME PRINTED	
			7-SEP-88 08:00	
LAYOUT	**OVERLAY**	**PAPER TYPE**	**FILE TYPE**	**COPIES**
P132D	CONDENSED	3_HOLE	DEFAULT	0001

From: SSDEV0::TEDONE "Matt DTN:522-2254 CX01-2/N26 01-Sep-1988 1532" 1-SEP-1988 17:33
To: GWYNED::GANESAN
Subj: KDB D proc listing


```
                        ;.SBTTL CONDITIONAL ASSEMBLY EQUATES

                        ;;******************************************************************************
                        ;
                        ;       EQUATES FOR CONDITIONAL ASSEMBLY CODE
                        ;
                        ;;******************************************************************************

000002                  C$TYPE  = 2             ;CONTROLLER TYPE; 0 = UDA, 1 = KDA, 2 = KDB
000000                      $$UDA       = 0
000001                      $$KDA       = 1
000002                      $$KDB       = 2
000000                  LAB     = 0             ;LAB LOOP ON TEST CONTINOUSLY = 1, NOT FOR LAB USE = 0
000000                  WRAP    = 0             ;WRAP TEST = 1 (LAB MUST = 1), NO WRAP TEST = 0
000000                  COPY4   = 0             ;0 = NORMAL 2 COPY HEADER COMPARE, 1 = ALL 4 MUST MATCH
000003                  D.E.C.  = 3
000001                  QDADM   = 1             ;0 = NO DM IN KDA, 1 = DM IN KDA
000001                  QREVB   = 1             ;0 = KDA PRIOR TO REV B, 1 = KDA REV B OR AFTER
000000                  SIMTST  = 0             ;0 = NORMAL CODE, 1 = ASSEMBLE SIMULATE TEST (KDB)
```

```
.PAGE
; UDA CONTROL MEMORY
;       47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32
;      H---+--------H--------+--H-----+--+--H-----------+
;      !CRY!   AD   !   AF   !   AS   !AB!CE!     RB    !
;      +---O--------O--------O--------O--+--+--O--------+
;
;       31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
;      +------------H--------+--H-----------H-----------+
;      !            !        !  !    IOC     !  RA/[BS]  !
;      !     TEST   !   BD   ! @!------------+-----------!
;      !            !        !  !   LITERAL IF AB= 1     !
;      +---O--------O--------O--+-----O--------O--------+
;
;       15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
;      H---+--+-----H----------H----------H-------------+
;      ! S !EN!K !P !         BRANCH ADDRESS (BA)        !
;      +---O--+-----O--------O--------O--------O--------+
; ALU    CRY - AF= 0-2 CARRY INTO ALU
;             - AF= 3-7 HIGH BYTE LITERAL SELECT
;        AD  - ALU DESTINATION   *ALU -> DB IF AB= 1*    *DB= DATA BUS*
;                0= ALU -> Q,DB            4= ALU/2 -> RB, Q/2 -> Q, ALU -> DB
;                1= ALU -> DB              5= ALU/2 -> RB, ALU -> DB
;                2= ALU -> RB, RA -> DB    6= ALUx2 -> RB, Qx2 -> Q, ALU -> DB
;                3= ALU -> RB,DB           7= ALUx2 -> RB, ALU -> DB
;        AF  - ALU FUNCTION   *SELECT HIGH BYTE LITERAL WITH AF= 3-7 AND CRY= 1*
;                0= ADD  R+S      2= REVSUB  R-S   4=  R AND S      6= R XOR  S
;                1= SUB  S-R      3= R OR S        5=  NOT R AND S  7= R XNOR S
;        AS  - ALU SOURCE= R,S *CHOOSE RA OR BS OR LITERAL AS SOURCE*
;                0= RA,Q          2= 0,Q           4= 0,RA          6= D,Q
;                1= RA,RB         3= 0,RB          5= D,RA (RA=RB) 7= D,0 (RA=RB)
; BUS    AB  - ALU TO DATA BUS; 0= BS SRC ON BUS, 1= ALU AS BUS SRC,ENB LITERAL
;        RB  - B PORT REGISTER, ALSO A PORT IF AS= 5 OR 7
;        BD  - BUS DESTINATION
;        RA  - A PORT REGISTER   *[AB= 0/1]*
;       [BS] - BUS SOURCE        * IF AB= 0*
; IOC    @   - IOC ENABLE        0= DISABLE       1= ENABLE
;        IOC - INPUT OUTPUT CONTROL SELECT (SPECIAL OPERATION)
;        LIT - 8 BIT LITERAL TO LOW  BYTE IF CRY= 0 AND AF= 0-2
;              8 BIT LITERAL TO HIGH BYTE IF CRY= 1 AND AF= 3-7
; BRANCH TEST
;        S   - TEST SENSE:  0= JUMP IF TEST LOW (-),  1= JUMP IF TEST HIGH (+)
;        TEST- TEST CONDITION SELECT                ,  ELSE  CONTINUE  TO  PC+1
;        EN  - ENABLE CONDITIONAL EXECUTE PER CE BIT 36  0= DISABLE, 1= ENABLE
;        CE  - CONDITIONAL ALU EXECUTE ON TEST RESULT
;              0= EXECUTE IF TEST FALSE (FAILS), 1= EXECUTE IF TEST TRUE(PASS)
;        K   - JUMP OPERATION
;                0= JUMP          1 & BA# OFFFH = JUMP   TO   SUBROUTINE
;                                 1 & BA= OFFFH = RETURN FROM SUBROUTINE
;        P   - PARITY BIT FOR ODD PARITY
;        BA  - BRANCH ADDRESS
.PAGE
```

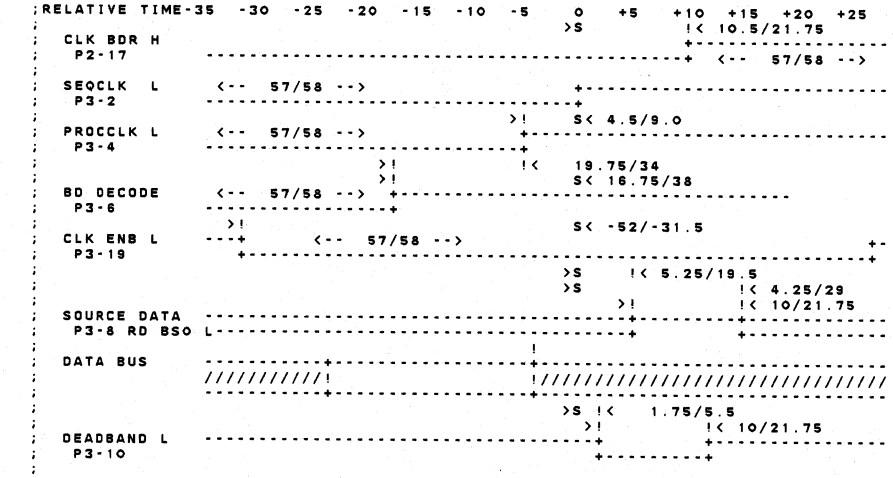


```
; INSTRUCTION FORMAT;
;       LABEL:  ALUOP\mod  intsrc\mod,dst\mod,bd  @ioc  %JUMPOP test,address
;       LABEL:  ALUOP\mod  (extsrc)  ,dst\mod,bd  @ioc  %JUMPTEST address
;               intsrc= 2901 internal source   RA
;               extsrc=      external source  (BS)
;               dst= 2901 internal destination RB
;               bd = 2901 external destination BD
;               mod: \F= execute false     \T= execute on true test condition
;                    \0= output RA         \N= no write to ALU
;                    \L= rotate ALU left   \LQ= shift Q left
;                    \R= rotate ALU right \RQ shift Q right
;               #LIT= literal source. Can be like ##LIT    ex. <bit00+bit04>
;               #LIT=  lower 8 bit literal with upper 8 bits = zero.
;               ##LIT= upper 8 bit literal with lower 8 bits = zero
; TYPES;        ALU INSTRUCTION FORMAT    SOURCE MODE     OPERATION
;       LABEL:  ALUOP RA,    RB,BD @IOC   ; register       RA OP RB-> RB
;       LABEL:  ALUOP (BS), RB,BD @IOC    ; data bus      (BS)OP RB-> RB, BS->BD
;       LABEL:  ALUOP # LIT,RB,BD @IOC    ; lo/hi literal # LIT OP RB->RB,BD
;       LABEL:  ALUOP ##LIT,RB,BD @IOC    ; upper literal ##LIT OP RB->RB,BD
; MNEMONICS;
;ALUOP= ADD      ADDC     AND      BIC      BIS      BIT      CLR      CMP
;       COM      COMB     DEC      DECB     DECC     INC      INCB     MOV
;       MOVB     NEG      NEGB     NOP      OR       ROT      ROTB     RSUB
;       SUB      SUBC     SHF      SHFB     TST      TSTB     XOR      XNOR
;JUMPOP= JUMP, ADDRESS;            JMP      J        J-
;        CALL, ADDRESS;            JSR      CALL     C        C-
;        RETURN;                   RET      RTN      R        R-
;        TEST (.+1 ALWAYS);        T        T-
;       AZTEC ONLY=      RTNI     JMPI     INT
; INNER INSTRUCTION TIMING RELATIONSHIP:
;       The signal SEQ CLK L determines the start of one instruction from
;       another.  All timing is referanced to this signal.  Branch conditions
;       are tested at the begining of the instruction.  The results of ALU and
;       bus operations are loaded into internal or external registers near the
;       end of the instruction.  The INTERFACE OUTPUT CONTROL (IOC) signals
;       are changed just a little prior to the the register loads.  The
;       signal UPROC H defines which processor is active during the present
;       instruction.  UPROC H = 1 (UPROC L = 0) means the UNIBUS processor is
;       active.  UPROC H = 0 (UPROC L = 1) means the DRIVE processor is
;       active.  A simplified timing relationship is shown below.
;
;       SEQ CLK L        ---+      +-----------+      +-----------+      +--
;                           +------+           +------+           +------+
; UPROC H          ----------------+                  +----------------+
;                                  +------------------+                +------
;  INSTRUCTION     ----------------+------------------+----------------+------
;  REGISTER           UNIBUS PROC  !   DRIVE PROC     !   UNIBUS PROC  !
;  INSTRUCTION     ----------------+------------------+----------------+------
;                                  ^              ^  ^
;                                  !              !  +-- LOAD REGISTERS
;                                  !              +----- CHANGE IOC'S
;                                  +------------------- TEST BRANCH CONDITION
.PAGE
```

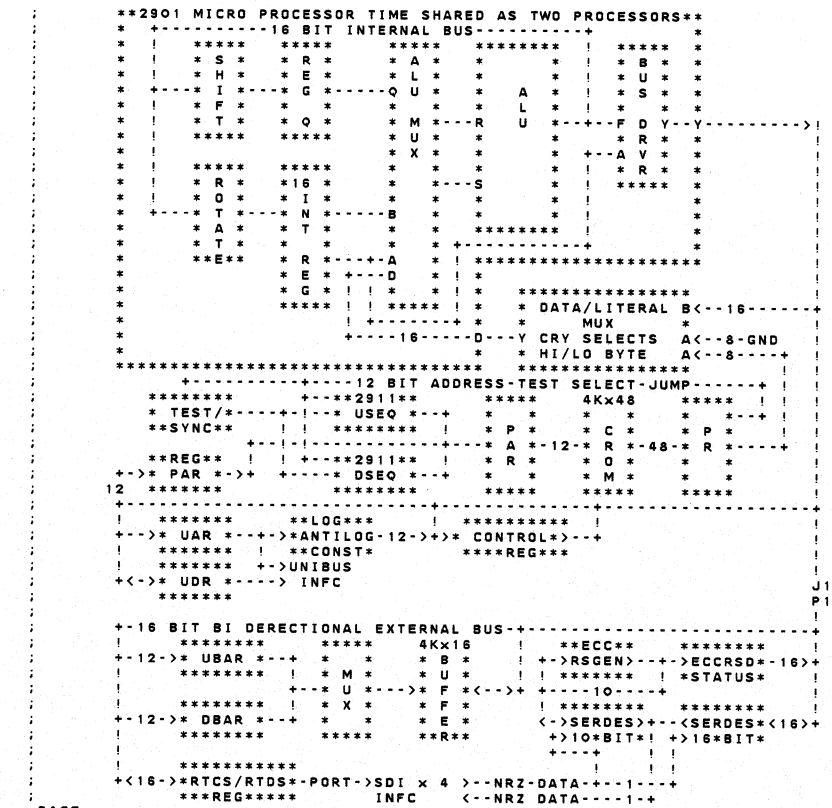
```
; UNIBUS DISK ADAPTER (UDA) TIMING
;
;                 ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++
;                 !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !!
;       CLOCK H  -""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-
;
;       F/F 1 H  --+58+-116-+  +-----+  +-----+  +-----+  +-----+  +-----+  +-
;                  +--+     +--+     +--+     +--+     +--+     +--+     +--+
;
;       F/F 2 H  --+     +--+     +--+     +--+     +--+     +--+     +--+
;       CLK ENB H  +-----+58+-116-+  +-----+  +-----+  +-----+  +-----+  +----
;
;                --+     +--+     +--+     +--+     +--+     +--+     +--+
;(K.SDI CLOCK IN H)+-----+58+-116-+  +-----+  +-----+  +-----+  +-----+  +----
;
;                  +-----+58+-116-+  +-----+  +-----+  +-----+  +-----+  +----
;       SEQ CLK L -+     +--+     +--+     +--+     +--+     +--+     +--+
;
;                  +--174---+        +--------+        +--------+        +----
;       UPROC  H --+        +--174---+        +--------+        +--------+
;
;                --+--------+--------+--------+--------+--------+--------+----
;       SEQ ADR H  !/////U+1!/////D+1!/////U+2!/////D+2!/////U+3!/////D+3!
;                --+--------+--------+--------+--------+--------+--------+----
;
;       PIPELINE --+--------+--------+--------+--------+--------+--------+----
;       ADDRESS REG!    D   !  U+1   !  D+1   !  U+2   !  D+2   !  U+3   !
;       PADR   H --+--------+--------+--------+--------+--------+--------+----
;
;       PIPELINE --+--------+--------+--------+--------+--------+--------+----
;       REG:CURRENT!    U   !   D    !  U+1   !  D+1   !  U+2   !  D+2   !
;       INSTRUCTION+--------+--------+--------+--------+--------+--------+----
;
.PAGE
```



```
; RELATIVE CLOCK TIMING - GENERAL
;
; Common K.SDI, K.DISK, K.???, UDA Terminology
;       1. Prefix all clocks with CLK, all bus sources with RD, all bus
;          destinations with LD.
;       2. Have common page order ie. ALU, CLOCKING, BRANCH, SEQ, PAR, CROM etc.
;
;K.DISK= CLOCK IN H-+                   +---------+                   +-------
;UDA=    CLK ENB H  +-------------------+         +-------------------+
;                       !                             !
;                   -+                   +---------+                   +------
;       CLK ENB1 H   +-------------------+         +-------------------+
;                       !                             !
;                   --+                   +---------+                   +-----
;       CLK ENB2 H    +-------------------+         +-------------------+
;                       !                             !
;                   ---+                   +---------+                   +----
;       CLK ENB3 H     +-------------------+         +-------------------+
;                       !                             !
;                   ----+                   +---------+                   +---
;       CLK ENB4 H      +-------------------+         +-------------------+
;                       !                             !
;                   -----+                   +---------+                   +--
;       CLK ENB5 H       +-------------------+         +-------------------+
;                       !                             !
;                       +-------------------+         +-------------------+
;       SEQ CLK L ------+                   +---------+                   +---
;                       !                             !
;                        +-----------------------------+
;       UPROC H   -------+                             +----------------------
;                       !                             !
;                                           +-----------------------------+
;       DLY UPROC H ------------------------+                             +---
;                       !                             !
;                      +-------------------+         +-------------------+
;       PROCCLK L -----+                   +---------+                   +----
;                       !                             !
;                     +-------------------+         +-------------------+
;       LD CLK L  ----+                   +---------+                   +-----
;                       !                             !
;                     +-------------------+         +-------------------+
;       BUFF WE   ----+                   +---------+                   +-----
;                       !                             !
;                        +-------------------+         +-------------------+
;       CLK BDR H -------+                   +---------+                   +--
;                       !                             !
;                 ------+-+---------------------------+-+---------------------
;                       !/!                           !/!
;       RD CLK L  ------+-+---------------------------+-+---------------------
;
.PAGE
```



.PAGE



.PAGE

LSCS FORM=QUAD

```
        .SBTTL  KDB MICROCODE
        .SBTTL  HARDWARE/SOFTWARE DEFINITIONS AND EQUATES

        ; *---------------------------------------------------------------------------*
        ; *      HARDWARE/SOFTWARE DEFINITIONS AND EQUATES                            *
        ; *      14-JAN-1985      UDA00.UQA                                           *
        ; *---------------------------------------------------------------------------*
        ;
        ;REVISION HISTORY:
        ;
```



```
;        PAGE ADDRESS     ADR*6=    WAS     SHOULD BE   TYPE          DESCRIPTION
;                         PATCH                         CHANGE
;               (OCT)     (OCT)    (OCT)      (OCT)     PERM TEMP
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
;       !     !         !         !         !           !     !     !                             !
;       +-----+---------+---------+---------+-----------+-----+-----+-----------------------------+
```


LSCS FORM=QUAD

```
;*------------------------------------------------------------------------*
;*     USING THE 2901 ALU AS A COMPARITOR                                 *
;*------------------------------------------------------------------------*
;
;                        LOGICAL OPERATIONS
;  FUNCTION  OPERATION          TEST CONDITION              RESULT
;   RA = 0      MOV      RA,RB   ZRO =    TRUE            RB= RA
;   RA # 0      MOV      RA,RB   ZRO =    FALSE             '
;   RA = RB     XOR      RA,RB   ZRO =    TRUE            RB= RA XOR  RB
;   RA # RB     XNOR     RA,RB   ZRO =    FALSE           RB= RA XNOR RB
;                        ARITHMETIC OPERATIONS - CARRY IN OFF
;   SAME SIGN   XOR      RA,RB\N NEG = TRUE       -    RA XOR RB -> BD  P-P,N-N
;   DIFF SIGN   XOR      RA,RB\N NEG = FALSE      -    RA XOR RB -> BD  P-N,P-N
;                                P-P,N-N  P-N,N-P
;   RA < RB     SUBC     RA,RB   CRY = TRUE   TRUE   RB-RA-1-> RB,BD  (BORROW)
;   RA>= RB     SUBC     RA,RB   CRY = FALSE (TRUE)         '
;   RA > RB    (RSUBC)   RA,RB   CRY = TRUE   FALSE  RA-RB-1-> RB,BD  (BORROW)
;   RA<= RB    (RSUBC)   RA,RB   CRY = FALSE  TRUE                '
;                        ARITHMETIC OPERATIONS - CARRY IN ON
;                                P-P,N-N  P-N,N-P
;   RA<= RB     SUB      RA,RB   CRY = TRUE   FALSE  RB-RA ->  RB,BD  (BORROW)
;   RA > RB     SUB      RA,RB   CRY = FALSE  TRUE           '
;   RA = RB     SUB      RA,RB   ZRO = TRUE   TRUE   RB-RA ->  RB,BD   (BORROW)
;   RA # RB     SUB      RA,RB   ZRO = FALSE  FALSE          '
;                                P-P,N-N  P-N,N-P
;   RA>= RB     RSUB     RA,RB   CRY = TRUE   FALSE  RA-RB ->  RB,BD   (BORROW)
;   RA < RB     RSUB     RA,RB   CRY = FALSE  TRUE           '
;   RA>= RB     CMP      RA,RB   CRY = TRUE   FALSE  RA-RB ->      BD   (BORROW)
;   RA < RB     CMP      RA,RB   CRY = FALSE  TRUE           '
;   RA = RB     RSUB     RA,RB   ZRO = TRUE   TRUE   RA-RB ->  RB,BD   (BORROW)
;   RA # RB     RSUB     RA,RB   ZRO = FALSE  FALSE          '
;
;  NOTE:     ADDITION  CRY = TRUE   MEANS ADD    ONE TO   NEXT HIGHER 16 BITS
;          SUBTRACTION CRY = FALSE  MEANS BORROW ONE FROM NEXT HIGHER 16 BITS
;              RA= REGISTER A SOURCE   RB= REGISTER B SOURCE,DESTINATION
;              RB\N=   REGISTER B SOURCE, NO LOAD
;              P-P= POSITIVE MINUS POSITIVE    P-N= POSITIVE MINUS NEGATIVE
;              N-N= NEGATIVE MINUS NEGATIVE    N-P= NEGATIVE MINUS POSITIVE
;          RA>= RB <==> RB<= RA          RA < RB <==> RB > RA
;
;  COMPARITOR EXCEPTIONS;
;      1. MAXIMUM POSITIVE NUMBER IS 32767 (HIGHER CHANGES THE SIGN BIT)
;      2. DIFFERENT SIGNS IMPLIES AN EQUATE IS NOT POSSIBLE EXCEPT
;         FOR ZERO SINCE IT HAS THE SAME VALUE POSITIVE OR NEGATIVE.
```



```
;*------------------------------------------------------------------------*
;*     PROCESSOR RESTRICTIONS - SUMMARY                                   *
;*------------------------------------------------------------------------*
;      PAR REGISTER - When loading the PAR register, neither source may be
;      changed to allow the address to be stable to the very end of the cycle.
;      2. Loading the PAR register must be unconditional to allow the pull-up
;      resistors enough time to get above the threshold voltage.
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *      TEST CONDITIONS FOR "U" PROCESSOR                                  *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;       THE TEST CONDITION EQUATES ARE OF THE FOLLOWING FORM:
                                ;               BIT4    POSITIVE TRUE CONDITION 0= FALSE, 1= TRUE (CROM15)
                                ;               BIT4    NEGATIVE TRUE CONDITION 1= FALSE, 0= TRUE (CROM15)
                                ;               BIT0-3  TEST CONDITION SELECT (CROM28-31)
                                ;       DEFINITIONS ARE IN OCTAL UNLESS OTHERWISE SPECIFIED
                                ; *-------------------------------------------------------------------------*
                                ; *              BUS PROCESSOR NEGATIVE TRUE CONDITION DEFINITION           *
                                ; *-------------------------------------------------------------------------*

000000                          UMPN    := 0            ; xDA - JUMP NOT = CONTINUE
000001                          NZRO    := 1            ; ALU - NOT = ZERO (NOT EQUALS)
000001                          NEQ     := 1            ; ALU - SRC NOT = DST
000002                          NCRY    := 2            ; ALU - CARRY OUT FALSE
000002                          LESS    := 2            ; ALU - SRC IS LESS THAN DST
000003                          NMSB    := 3            ; ALU - MSB =0
000003                          NNEG    := 3            ; ALU - COMPARITOR MSB = 0,NOT NEGATIVE (POS OR ZERO)
000004                          LSB     := 4            ; ALU - LSB =1
000005                          TEST    := 5            ; xDA - EXT.TEST L (P3-3) ACTIVE, DO EXTRA TEST CODE
000006                          NUPF    := 6            ; xDA - FLAG RESET BY IOC OF U PROCESSOR @RUPF
000006                          DPROC   := 6            ; xDA - SPLIT ; @RUPF -> DPROC JUMP/UPROC CONTINUE (PC+1)
000007                          ACLO    := 7            ; xDA - AC POWER FAILING
000007                          nrpok   := 7    ;temp   ; xDA - not power ok [qda compat]
000010                          FTEST   := 10           ; xDA - FAST SELF TEST
000011                          STOP    := 11           ; xDA - STOP COMMAND ISSUED - NO MORE BI COMMANDS
000012                          CPE     := 12           ; xDA - CONTROL MEMORY PARITY ERROR
000013                          ETST1   := 13           ; xDA - EXTERNAL TEST 1
000014                          MERR    := 14           ; xDA - BI COMMAND ERROR ON MASTER PORT (DURING XFER)
000015                          SCAN    := 15           ; xDA - IP ACCESSED
000015                          POLL    := SCAN
000016                          CDONE   := 16           ; xDA - COMMAND DONE
000017                          NBIBAD  := 17           ; xDA - BI BAD L SIGNAL NOT ASSERTED LOW (TBAD H IS LOW)
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *              BUS PROCESSOR POSITIVE TRUE CONDITION DEFINITION           *
                                ; *-------------------------------------------------------------------------*

000020                          UMP     := 20           ; xDA - JUMP UNCONDITIONAL
                                ;JMP    := 20           ; ASSEMBLER DEFINED - JMP IS UNCONDITIONAL JUMP
000021                          ZRO     := 21           ; ALU - EQUALS ZERO
000021                          EQ      := 21           ; ALU - SRC = DST
000022                          CRY     := 22           ; ALU - CARRY OUT TRUE
000022                          GTE     := 22           ; ALU - SRC IS GREATER THAN OR SAME AS DST
000023                          MSB     := 23           ; ALU - MSB = 1
000023                          NEG     := 23           ; ALU - RESULT NEGATIVE          MSB= 1
000024                          NLSB    := 24           ; ALU - LSB = 0
000025                          NTEST   := 25           ; xDA - EXT.TEST L (P3-3) INACTIVE
000026                          UPF     := 26           ; xDA - FLAG SET BY IOC OF BUS PROCESSOR @SUPF
000026                          UPROC   := 26           ;       @SUPF -> UPROC JUMP / DPROC CONTINUE (PC+1)
000027                          NACLO   := 27           ; xDA - AC POWER NOT FAILING
000027                          rpok    := 27   ;temp   ; xDA - power ok [qda compat]
000030                          NFTEST  := 30           ; xDA - SLOW SELF TEST
000030                          STEST   := NFTEST
000031                          NSTOP   := 31           ; xDA - STOP COMMAND NO ISSUED - MORE BI COMMANDS
000032                          NCPE    := 32           ; xDA - NO CONTROL MEMORY PARITY ERROR
000033                          NETST1  := 33           ; xDA - NOT EXTERNAL TEST 1
000034                          NMERR   := 34           ; xDA - NO BI COMMAND ERROR ON MASTER PORT (DURING XFER)
000035                          NSCAN   := 35           ; xDA - NO IP ACCESS
000035                          NPOLL   := NSCAN
000036                          NCDONE  := 36           ; xDA - NOT COMMAND DONE
000037                          BIBAD   := 37           ; xDA - BI BAD L IS ASSERTED LOW (TBAD H IS HIGH)
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *     TEST CONDITIONS FOR "D" PROCESSOR (WHERE DIFFERENT ONLY)            *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *              DRIVE PROCESSOR NEGATIVE TRUE CONDITION DEFINITION         *
                                ; *-------------------------------------------------------------------------*

000006                          NDPF    := 6            ; xDA - D PROCESSOR FLAG NOT SET (@RDPF)
000010                          ndclk   := 10   ;temp   ; sdi - serdes parallel data not ready
000010                          NPRDY   := 10           ; SDI - SERDES PARALLEL DATA NOT READY
000011                          NCSR    := 11           ; SDI - CONTROL STATE NOT READY - RTCS NOT READY TO UPDATE (RELOAD)
000012                          SECTR   := 12           ; SDI - SECTOR PULSE
000012                          nrrdy   := 12   ;temp   ; serdes - ecc residue parallel data not ready [qda compat]
000013                          RPE     := 13           ; xDA - RAM PARITY ERROR
000014                          NDSER   := 14           ; NOT REAL TIME DRIVE STATE ERROR
000015                          NDSR    := 15           ; DRIVE STATE NOT READY - NEW RTDS NOT RECEIVED SINCE LAST WRITTEN TO RTCS
000016                          nlate   := 16   ;temp   ; sdi - no serdes over/under runs
000016                          NOVER   := 16           ; SDI - NO SERDES OVER/UNDER RUNS
000017                          NWRC    := 17           ; xDA - WORD RATE CLOCK LOW

                                ; *-------------------------------------------------------------------------*
                                ; *              DRIVE PROCESSOR POSITIVE TRUE CONDITION DEFINITION         *
                                ; *-------------------------------------------------------------------------*

000026                          DPF     := 26           ; xDA - D PROCESSOR FLAG SET (@SUPF)
000030                          dclk    := 30   ;temp   ; sdi - serdes parallel data ready
000030                          PRDY    := 30           ; SDI - SERDES PARALLEL DATA READY
000031                          CSR     := 31           ; SDI - CONTROL STATE READY - RTCS READY TO UPDATE (RELOAD)
000032                          NSECTR  := 32           ; SDI - NO SECTOR PULSE
000032                          rrdy    := 32   ;temp   ; serdes - ecc residue parallel data ready [qda compat]
000033                          NRPE    := 33           ; xDA - RAM PARITY ERROR
000034                          DSER    := 34           ; SDI - REAL TIME DRIVE STATE ERROR (PARITY,PULSE, OR DATA)
000035                          DSR     := 35           ; DRIVE STATE READY - NEW RTDS RECEIVED SINCE LAST WRITTEN TO RTCS
000036                          late    := 36   ;temp   ; sdi - serdes over/under run has occurred
000036                          OVER    := 36           ; SDI - SERDES OVER/UNDER RUN HAS OCCURRED
000037                          WRC     := 37           ; xDA - WORD RATE CLOCK HIGH
```



```
                                ;       *---------------------------------------------------*
                                ;       *              INPUT/OUTPUT CONTROL REGISTER        *
                                ;       *---------------------------------------------------*
                                ;
                                ;       EACH PROCESSOR (BUS OR DRIVE) HAS AN ADDRESSABLE LATCH
                                ;       WHICH IS USED FOR INPUT/OUTPUT CONTROLS.  SOME BITS HAVE MULITPLE
                                ;       USES.  EACH SIGNAL IS INDIVIDUALLY SET OR RESET IF ENABLED BY THE
                                ;       CURRENT MICROINSTRUCTION.
                                ;               BIT3    0= SIGNAL TO LOW LEVEL, 1= SIGNAL TO HIGH (CROM23)
                                ;               BIT0-2  IOC SELECT (CROM20-22)
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *              BUS PROCESSOR IOC SIGNAL DEFINITIONS                       *
                                ; *-------------------------------------------------------------------------*

000000                          SUPF    := 0            ; xDA - UTEST H SET
000001                          GORD    := 1            ; xDA - GO_READ
000002                          LNPAG   := 2            ; xDA - LOAD NEXT PAGE ADDRESS LOWER
000003                          WRSND   := 3            ; xDA - WRITE SECOND BUFFER
                                ;LOFF   := 3            ; xDA - LOAD ADDRESS BUFFER OFFSET
000004                          RDFST   := 4            ; xDA - READ FIRST BUFFER
000005                          WRFST   := 5            ; xDA - WRITE FIRST BUFFER
000006                          LCOM    := 6            ; xDA - LOAD COMMAND
000007                          LADD    := 7            ; xDA - LOAD COMAND ADDRESS LOWER

000010                          RUPF    := 10           ; xDA - UTEST H CLR
                                ;SCLR   := 11           ; xDA - CLEAR CROM PE (defined in UDIAG.UQA)
                                ;RCLR   := 11   ;temp   ; xDA - CLEAR CROM PE (defined in UDIAG.UQA) [qda compat]
                                ;WRBYT  := 12           ; xDA - WRITE BYTE ON NEXT OPERATION
000012                          GOWR    := 12           ; xDA - GO WRITE
000013                          RBCAI   := 13           ; xDA - RESET BCAI
000014                          RDNXT   := 14           ; xDA - READ NEXT BUFFER
000015                          WRNXT   := 15           ; xDA - WRITE NEXT BUFFER
000016                          LBWR    := 16           ; xDA - WRITE LAST BUFFER AND GO_WRITE
000017                          HADD    := 17           ; xDA - ADDRESS
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *              DRIVE PROCESSOR IOC SIGNAL DEFINITIONS                     *
                                ; *-------------------------------------------------------------------------*
                                ;       CERTAIN SIGNALS MUST BE SEQUENCED IN ORDER FOR THE HARDWARE TO WORK:
                                ;       ?       WRITING DATA            FIRST @RCMD     THEN @SSE
                                ;       ?       WRITING COMMANDS        FIRST @SCMD     THEN @SSE
                                ;       ?       READING DATA            FIRST @RCMD     THEN @SSE
                                ;       ?       READING COMMANDS        FIRST @RCMD     THEN @SSE
                                ;       ?       DEFAULT                 FIRST @SCMD     THEN @SSE
                                ;

000000                          SDPF    := 0            ; xDA - SET DRIVE PROCESSOR FLAG (TEST AS COMPLIMENT)
000001                          RRSGEN  := 1            ; SDI - RESET RSGEN CLOCK. THIS WILL ALLOW RSGEN CLOCKS TO OCCUR
                                ;                       ;       FOLLOWING A @SRSGEN A @RRSGEN WILL UPDATE RSGEN AND ECC REGISTERS
000002                          SCMD    := 2            ; SDI - SET SDI LEVEL 2 COMMAND
                                ;SCLR   := 3            ; SDI - SET SDI INFC I/O CLEAR (defined in DDIAG.UQA)
000004                          RSE     := 4            ; SERDES - RESET SERDES ENABLE/-RESET REG,CNT   MRO,CE
000005                          RRM     := 5            ; SERDES - RESET READ MODE                      SDIC,POM
000005                          SWM     := 5            ; SERDES - SET WRITE MODE
000006                          RECC    := 6            ; SERDES - RESET ECC ENABLE
000007                          RECCT   := 7            ; SERDES - RESET ECC TIME

000010                          RDPF    := 10           ; xDA - RESET D PROCESSOR FLAG (TEST AS COMPLIMENT)
000011                          SRSGEN  := 11           ; SDI - SET RSGEN CLOCK. THIS WILL FORCE RSGEN CLOCK TO BE LOW
000012                          RCMD    := 12           ; SDI - FOR ALL OTHER FUNCTIONS
                                ;RCLR   := 13           ; SDI - RESET SDI INFC I/O CLEAR (defined in DDIAG.UQA)
000014                          SSE     := 14           ; SERDES - SET SERDES ENABLE/-RESET REG,CNT     MRO,CE
000015                          SRM     := 15           ; SERDES - SET READ MODE
000015                          RWM     := 15           ; SERDES - RESET WRITE MODE
000016                          SECC    := 16           ; SERDES - SET ECC ENABLE
000017                          SECCT   := 17           ; SERDES - SET ECC TIME
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *                    2901 INTERNAL REGISTER SUMMARY                       *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ; These are registers which are internal to the 2901 bit slice chip. They
                                ; consist of 16 Dual Port RAM registers and 1 accumulator (Q) register. The 16
                                ; RAM registers are configured for a 16 bit rotate while the Q register is a 16
                                ; bit shift.
                                ;
                                ;MNEMONIC        DESCRIPTION                                             UPROC   DPROC
                                ;       OCTAL ADDRESS
                                ;
                                ;       := Q    ; BUS PROCESSOR TIMER/COUNTER REGISTER  R/W              -
                                ;       := R0   ; BUS PROCESSOR TEMPORARY REGISTER                      R/W     -
                                ;       := R1   ; BUS PROCESSOR TEMPORARY REGISTER                      R/W     -
                                ;       := R2   ; BUS PROCESSOR TEMPORARY REGISTER                      R/W     -
                                ;       := R3   ; BUS PROCESSOR TEMPORARY REGISTER                      R/W     -
000004                          RLL     := R4   ; RICHY LARY'S LOCK AND SYSTEM STATUS                   R/W     R/W
000004                          CRI     := R4   ; *** TEMP UNTIL MATT'S CODE CHANGED
000005                          UBAR    := R5   ; BUS PROCESSOR BUFFER ADDRESS REG IMAGE                R/W     -
000006                          IUAR    := R6   ; BUS ADDRESS REGISTER IMAGE                            R/W     -
000007                          UER     := R7   ; UPROC ERROR REGISTER                                  R/W     -
000010                          UTMP    := R10  ; UPROC TEMPORARY REGISTER                              R/W     -
000011                          DER     := R11  ; DPROC ERROR REGISTER                                  -       R/W
                                ;       := R11  ; DRIVE PROCESSOR TIMER/COUNTER REGISTER                -       R/W
000012                          DTMP    := R12  ; DPROC TEMPORARY REGISTER                              -       R/W
000012                          RPC     := R12  ; DM INTERPRETER PC                                     -       R/W
                                ;       := R12  ; DRIVE PROCESSOR TEMPORARY REGISTER                    -       R/W
000013                          RIB     := R13  ; DM INTERPRETER INSTRUCTION BUFFER                     -       R/W
                                ;       := R13  ; DRIVE PROCESSOR TEMPORARY REGISTER                    -       R/W
000014                          RCC     := R14  ; DM INTERPRETER ARITH/LOGICAL CONDITION CODE           -       R/W
                                ;       := R14  ; DRIVE PROCESSOR TEMPORARY REGISTER                    -       R/W
000015                          DBAR    := R15  ; DRIVE PROCESSOR BUFFER ADDRESS REGISTER IMAGE         -       R/W
000015                          RCY     := R15  ; DM INTERPRETER LAST ARITH OP CARRY                    -       R/W
000016                          RT1     := R16  ; DM INTERPRETER GENERAL PURPOSE TEMP                   -       R/W
                                ;       := R16  ; DRIVE PROCESSOR TEMPORARY REGISTER                    -       R/W
                                ;       := R17  ; DRIVE PROCESSOR TEMPORARY REGISTER                    -       R/W

                                ; *-------------------------------------------------------------------------*
                                ; *                2901 INTERNAL REGISTER SUMMARY - DIAGNOSTICS             *
                                ; *-------------------------------------------------------------------------*

                                ;MNEMONIC        DESCRIPTION                                             UPROC   DPROC
                                ;       OCTAL ADDRESS

000000                          TSTCNT  := Q    ; BUS PROCESSOR TIMER/COUNTER REGISTER                  R/W     -
000003                          HANG    := R3   ; BUS PROCESSOR TEMPORARY REGISTER                      R/W     -
000010                          UDDI    := R10  ; BUS DATA REGISTER REGISTER IMAGE                      R/W     -
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *                  2901 EXTERNAL HARDWARE REGISTER SUMMARY                *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;MNEMONIC        DESCRIPTION
                                ;                                #1= On xDA#1 PC BOARD.
                                ;      OCTAL ADDRESS             #2= On xDA#2 PC BOARD.                  UPROC    DPROC

000000                          NOPREG  := 0     ; (NON EXISTANT)                                        WRITE    WRITE
000002                          RTDS    := 2     ; NO OPERATION REGISTER (NON EXISTANT)                  (RD)     READ
000002                          PAR     := 2     ; PIPELINE ADDRESS REGISTER                             WRITE#1  WRITE #1
000003                          BUF     := 3     ; RAM DATA BUFFER                                       R/W #2   R/W #2
000004                          CR      := 4     ; DPROC - CONTROL REGISTER SOURCE                        -       R/W #2
                                                 ; UPROC - CONTROL REGISTER DESTINATION                  WRITE#1
000014                          UCRS    := 14    ; UPROC - CONTROL REGISTER SOURCE                       READ#1   (RD)
000004                          UCRD    := 4     ; UPROC - CONTROL REGISTER DESTINATION                  WRITE#1
000001                          DCRS    := 1     ; DPROC - CONTROL REGISTER SOURCE                                READ#2
000004                          DCRD    := 4     ; DPROC - CONTROL REGISTER DESTINATION                           WRITE#2
000005                          RTCS    := 5     ; DPROC - REAL TIME CONTROLLER STATE                     -       WRT#2
000015                          BCAIS   := 15    ; UPROC - BCAI REGISTER SOURCE                          READ #1  (RD)
000005                          BCAID   := 5     ; UPROC - BCAI REGISTER DESTINATION                     WRITE#1  (RD)
000015                          UDS     := 15    ; UPROC - BUS DATA REGISTER SOURCE      [qda compat]    READ #1  (RD)
000005                          UDD     := 5     ; UPROC - BUS DATA REGISTER DESTINATION [qda compat]    WRITE#1  (RD)
000006                          BREG    := 6     ; UPROC - BI REGISTER                                   WRITE #1
000006                          SD      := 6     ; DPROC - SERDES 16 DATA REGISTER                       (RD)     R/W #2
000007                          ECC     := 7     ; DPROC - RS GENERATOR REGISTER (RESIDUE)               (RD)     READ#2
000007                          BAR     := 7     ; UPROC - BUFFER ADDRESS REGISTER                       WRITE#2  -
                                                 ; DPROC - BUFFER ADDRESS REGISTER                        -       WRT #2
000010                          FLOAT   := 10    ; NO BUS SOURCE REGISTER & BUS FLOAT                    READ     READ
                                ; *-------------------------------------------------------------------------*
                                ; *      BIT DEFINITIONS                                                    *
                                ; *-------------------------------------------------------------------------*

100000                          BIT15   := 100000
040000                          BIT14   := 40000
020000                          BIT13   := 20000
010000                          BIT12   := 10000
004000                          BIT11   := 4000
002000                          BIT10   := 2000
001000                          BIT09   := 1000
000400                          BIT08   := 400
000200                          BIT07   := 200
000100                          BIT06   := 100
000040                          BIT05   := 40
000020                          BIT04   := 20
000010                          BIT03   := 10
000004                          BIT02   := 4
000002                          BIT01   := 2
000001                          BIT00   := 1
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *                  2901 INTERNAL REGISTER DEFINITIONS                     *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;MNEMONIC  ADDRESS       DESCRIPTION                              UPROC   DPROC
                                ;
                                ;RLL     := R4           ; RICHY LARY LOCK & STATUS REGISTER      R/W     R/W
                                ;
000001                          BLOCK   := BIT00         ; BUFFER LOCK FLAG, 0 = LOCKED, 1 = FREE
                                ;       := BIT01         ; BUFFER STACK POINTER
                                ;       := BIT02         ; BUFFER STACK POINTER
                                ;       := BIT03         ; BUFFER STACK POINTER
                                ;       := BIT04         ; BUFFER STACK POINTER
                                ;       := BIT05         ; BUFFER STACK POINTER
                                ;       := BIT06         ; BUFFER STACK POINTER
                                ;       := BIT07         ; BUFFER STACK POINTER
                                ;
                                ;       := BIT08         ; BUFFER STACK POINTER
001000                          INDIAG  := BIT09         ; 0 = FUNCTIONAL CODE, 1 = xDA DIAGNOSTIC CODE RUNNING
                                ;       := BIT10         ; NOT USED
004000                          CN.ERR  := BIT11         ; 0 = NO ACTION, 1 = D.PROC FATAL CONTROLLER ERROR (CODE IN R11)
010000                          DXFC    := BIT12         ; 0 = NO ACTION, 1 = U.PROC XFC SERVICE REQUIRED
020000                          DMBEG   := BIT13         ; 0 = NO ACTION, 1 = START DM INTERPRETER (START ADDRESS IN RPC)
040000                          DMODE   := BIT14         ; 0 = NORMAL MODE, 1 = DIAGNOSTIC MACHINE MODE
100000                          PLOCK   := BIT15         ; PROCESSOR LOCK FLAG, 0 = LOCKED, 1 = FREE
                                ;
                                ;UER     := R7           ; UPROC ERROR REGISTER                    R/W      -
                                ;DER     := R11          ; DPROC ERROR REGISTER                     -       R/W
                                ;
                                ;       THE ERROR DISPLAY IS BIT ENCODED TO CORRESPOND TO THE ERROR MESSAGE
                                ;
                                ;       *** SEE BUS I/O REGISTER DEFINITIONS    ***
                                ;
```

```
                        ; *-------------------------------------------------------------------------------*
                        ; *                        2901 EXTERNAL REGISTER DEFINITIONS                     *
                        ; *-------------------------------------------------------------------------------*
                        ;
                        ;MNEMONIC  ADDRESS       DESCRIPTION                                    UPROC   DPROC
                        ;
                        ;NOPREG  := 00           ; NO OPERATION REGISTER (NON EXISTANT)         R/W     R/W
                        ;
                        ;        := BIT00        ; + NON   EXISTANT
                        ;        := BIT01        ; +   '       '
                        ;        := BIT02        ; +   '       '
                        ;        := BIT03        ; +   '       '
                        ;          --------
                        ;        := BIT04        ; +   '       '
                        ;        := BIT05        ; +   '       '
                        ;        := BIT06        ; +   '       '
                        ;        := BIT07        ; +   '       '
                        ;          --------
                        ;        := BIT08        ; +   '       '
                        ;        := BIT09        ; +   '       '
                        ;        := BIT10        ; +   '       '
                        ;        := BIT11        ; +   '       '
                        ;          --------
                        ;        := BIT12        ; +   '       '
                        ;        := BIT13        ; +   '       '
                        ;        := BIT14        ; +   '       '
                        ;        := BIT15        ; +   '       '
```



```
                        ;MNEMONIC  ADDRESS       DESCRIPTION                                    UPROC   DPROC
                        ;
                        ;UCRS    := 14           ; UPROC CONTROL REGISTER SOURCE                READ      -
                        ;UCRD    := 04           ; UPROC CONTROL REGISTER DESTINATION           WRITE     -
                        ;
                        ;       THIS REGISTER IS READ/WRITE.  IT IS DIFFERENT FROM xDA#2 CR REGISTER
                        ;       THAT THE LED'S ARE ALWAYS ON. DIAGNOSTIC MODE HAS NO EFFECT ON ANY
                        ;       xDA#1 HARDWARE.
                        ;
                        ;        := BIT00        ; LSB WORD COUNT BIT
                        ;        := BIT01        ;
                        ;        := BIT02        ;
                        ;        := BIT03        ; MSB WORD COUNT BIT
                        ;          -------
                        ;        := BIT04        ; RESERVED
                        ;        := BIT05        ; RESERVED
000100                  U8K      := BIT06        ; 0 = ENABLE DPROC FAIL TEST, 1 = ENABLE UPROC FAIL TEST
                        ;DPM     := BIT07        ; DIAGNOSTIC MODE L  *THIS BIT DOES NOTHING TO THE HARDWARE xDA#1*
                        ;          -------
000400                  vaxint   := BIT08  ;temp ; 1 = vax kludge interrupt in progress [uda compat]
                        ;        := BIT09        ; NOT USED
002000                  DFAIL    := BIT10        ; 0 = ENABLE DPROC SEQUENCER,1= FORCE DPROC TO FAIL TEST
004000                  DRINIT   := BIT11        ; 1 = DRIVE INIT IN PROGRESS (USED FOR SOFTWARE INIT)
                        ;          -------
                        ;        := BIT12        ; LED 1 L, 0 = TURN LED ON, 1 = TURN LED OFF
                        ;        := BIT13        ; LED 2 L       "                "
                        ;        := BIT14        ; LED 4 L       "                "
                        ;        := BIT15        ; LED 8 L       "                "
                        ;
```

```
                     ;MNEMONIC  ADDRESS      DESCRIPTION                           UPROC   DPROC
                     ;
                     ;DCRS    := 01           ; DPROC CONTROL REGISTER SOURCE         -      READ
                     ;DCRD    := 04           ; DPROC CONTROL REGISTER DESTINATION    -      WRITE
                     ;
                     ;            NOTE - 1. IN DIAGNOSTIC MODE THE D PROCESSOR SHOULD MOVE CRI
                     ;            TO CR AND WAIT FOR THE LEAST SIGNIFICANT(some) BIT TO COME
                     ;            ON BEFORE PROCEEDING.  THIS WAY THE U PROCESSOR CAN ENABLE
                     ;            OR DISABLE RAM PARITY ERROR AND FLASH THE LIGHTS FOR
                     ;            BOTH BOARDS.  LEAST SIGNIFICANT BIT = 0 MEANS WAIT, = 1
                     ;            MEANS CONTINUE.
                     ;            2. IN DIAGNOSTIC MODE THERE IS A 1 BIT DELAY FROM DIAG DATA TO
                     ;            NRZ DATA IN AND A 21 BIT DELAY FROM LOADING RTCS TO VALID RTDS.
                     ;
                     ;        := BIT00        ; 1 = SELECT DRIVE PORT 1
                     ;        := BIT01        ; 1 = SELECT DRIVE PORT 2
                     ;        := BIT02        ; 1 = SELECT DRIVE PORT 3
                     ;        := BIT03        ; 1 = SELECT DRIVE PORT 4
                     ;           -------
000020               FDCLK    := BIT04        ; USE DIAGNOSTIC CLOCK=0,USE FAST CLOCK=1
000040               SERIAL   := BIT05        ; SERIAL PORT
000100               RAMPE    := BIT06        ; 0 = INHIBIT RAM PARITY ERROR, 1 = ENABLE RAM PARITY ERROR
000200               OPM      := BIT07        ; DIAGNOSTIC MODE L      0= ENABLE  OUTPUT BITS 08-15,
                     ;           -------                               1= DISABLE OUTPUT BITS 08-15
000400               ODDP     := BIT08        ; ODD PARITY H  BIT07= 0 & 0= EVEN RAM PARITY, 1= ODD PARITY
                     ;        := BIT09        ; UNUSED HARDWARE BIT
002000               DDC      := BIT10        ; DIAGNOSTIC DRIVE CLOCK
004000               DDD      := BIT11        ; DIAGNOSTIC DRIVE DATA
                     ;           -------
010000               LED1     := BIT12        ; LED 1 L, 0 = TURN LED ON, 1 = TURN LED OFF
020000               LED2     := BIT13        ; LED 2 L       "                 "
040000               LED4     := BIT14        ; LED 4 L       "                 "
100000               LED8     := BIT15        ; LED 8 L       "                 "
                     ;
170000               LEDS     := LED8+LED4+LED2+LED1
                     ;
```



```
;
;                              SDI INTERCONNECT SIGNALS
;
; ------------+                                                     +-------------
;             !-----------WRITE/COMMAND DATA   (WCD)------------!
; CONTROLLER  !-----REAL TIME CONTROLLER STATE FRAME (RTCS)---->!      DRIVE
;             !<----------READ/RESPONSE DATA   (RRD)------------!
;             !<----REAL TIME DRIVE STATE FRAME       (RTDS)-----!
; ------------+                                                     +-------------
;
;
;BIT 15 ! 0   1   2   3 ! 4   5   6   7 ! 8   9  10  11 !12  13  14  15 ! 0
;   -----+---------PREAMBLE--------------SYNC--------STATUS--------------+-----
;    ! P ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 1 !RDY!WRT!RD !INI!RES!RES!PTY! 0 !
;   -----+---------------------------------------------------------------+-----
;        +              !               !               !N/T N/T         +
;
;
;
;RTDS                          REAL TIME DRIVE STATE FRAME
;
;BIT 15 ! 0   1   2   3 ! 4   5   6   7 ! 8   9  10  11 !12  13  14  15 ! 0
;   -----+---------PREAMBLE--------------SYNC--------STATUS--------------+-----
;    ! P ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 1 !RDY!ATN!R/W!SEC!IDX!AVL!PTY! 0 !
;   -----+---------------------------------------------------------------+-----
;
;
; 1. EVEN PARITY OVER COMPLETE FRAME
; 2. DATA SENT LEAST SIGNIFICANT BIT FIRST
; 3. IN DIAGNOSTIC MODE IT TAKES 21 CLOCKS FROM LOADING RTCS UNTIL RTDS IS VALID
; 4. N/T= DESERIALIZER RECEIVES NON TRI-STATED AND TRI-STATED.
;
```

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;RTCS    := 05          ; REAL TIME CONTROLLER STATE                    -      WRITE
                                ;
                                ;       1.DO NOT CHANGE READ AND WRITE GATE IN THE SAME INSTRUCTION SINCE IT
                                ;         WOULD BE POSSIBLE FOR BOTH TO BE SENT ACTIVE.
                                ;       2.EACH TIME THIS REGISTER IS LOADED, ONE FRAME WILL BE SENT REGARDLESS
                                ;         OF THE STATE OF THE 'CONT H' BIT.
                                ;       3.BEFORE CHANGING PORTS, TURN OFF THE 'CONT H' BIT AND WAIT FOR 'RTCS
                                ;         READY' OF THE 'RTDS' REGISTER TO INDICATE THE TRANSMISSION HAS COMPLETED.
                                ;
000001                          CONT     := BIT00       ; +CONT H        0= SEND ZERO'S , 1= SEND FRAME
000002                          WRT      := BIT01       ; +WRT GATE H    0= NO WRITE GATE,1= ENABLE WRITTING OF DATA
000004                          RD       := BIT02       ; +RD GATE H     0= NO READ DATA, 1= SEND READ DATA
000010                          CRDY     := BIT03       ; +CON RCVR RDY H 0= NOT READY, 1= READY TO RECEIVE COMMAND/DATA
                                ;          -------
000020                          INI      := BIT04       ; +INIT H        0= NO INIT, 1= INITIALIZE (RESET) DRIVE
000040                          rtcs05   := bit05 ;temp ; +unused 1
000040                          RTCS13   := BIT05       ; +UNUSED 1      0= DEFAULT
000100                          rtcs06   := bit06 ;temp ; +unused 0
000100                          RTCS14   := BIT06       ; +UNUSED 0      0= DEFAULT
000200                          FPE      := BIT07       ; +FORCE PARITY ERROR H 0= SEND (CORRECT) EVEN PARITY,1= SEND ODD PARITY
                                ;          -------
                                ;        := BIT08       ; +RESERVED
                                ;        := BIT09       ; +      "
                                ;        := BIT10       ; +      "
                                ;        := BIT11       ; +      "
                                ;          --------
                                ;        := BIT12       ; +      "
                                ;        := BIT13       ; +      "
                                ;        := BIT14       ; +      "
                                ;        := BIT15       ; +RESERVED
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;RTDS    := 02          ; REAL TIME DRIVE STATE (NON EXISTANT)          -      READ
                                ;
                                ;       WHEN FIRST SELECTING A PORT, THE FIRST 'RTDS' FRAME MUST BE DISREGARDED.
                                ;       TO DO THIS, SEND ONLY ONE 'RTCS' FRAME THEN WAIT FOR 'FRAME SENT' TO
                                ;       GO ACTIVE
                                ;
000001                          DRDY     := BIT00       ; +D RCVR RDY H 0= DO NOT SEND,  1= RECEIVER READY FOR COMMAND
000002                          ATTN     := BIT01       ; +ATTN H        0= NO ERROR,    1=
                                ;        := BIT02       ; +RESERVED
                                ;        := BIT03       ; +      "
                                ;          --------
000020                          SEC      := BIT04       ; +SECTOR H      0= NO SECTOR    1= SECTOR PULSE(USE 1-0 EDGE)
000040                          IDX      := BIT05       ; +INDEX H       0= NO INDEX     1= INDEX PULSE (USE 1-0 EDGE)
000100                          rtds06   := bit06 ;temp ;
000100                          AVAIL    := BIT06       ; +AVAILABLE     0= NOT AVAILABLE 1= DEVICE AVAILABLE FOR USE
                                ;        := BIT07       ; +RESERVED
                                ;          --------
                                ;        := BIT08       ; +      "
                                ;        := BIT09       ; +      "
                                ;        := BIT10       ; +      "
                                ;        := BIT11       ; +      "
                                ;          --------
                                ;        := BIT12       ; +      "
                                ;        := BIT13       ; +      "
                                ;        := BIT14       ; +      "
100000                          RWRDY    := BIT15       ; +R/W RDY H     0= NOT R/W READY, 1= READ/WRITE READY
```

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                  UPROC   DPROC
                                ;
                                ;PAR     := 02          ; PROGRAM ADDRESS REGISTER                   WRITE   WRITE
                                ;
                                ;               A  WRITE  TO THIS REGISTER WILL CAUSE A JUMP TO AN ADDRESS
                                ;               BETWEEN FOOH AND FFF HEXIDECIMAL. THE INSTRUCTION JUMPED TO MUST
                                ;               CONTAIN A  JUMP  INSTRUCTION TO PREVENT A FALL BACK TO THE
                                ;               INSTRUCTION AFTER THE PAR UPDATE.
                                ;
                                ;                  15 +  NOT  IMPLEMENTED
                                ;                  14 +   '       '
                                ;                  13 +   '       '
                                ;                  12 +   '       '
                                ;                --------
                                ;                  11 +MEMORY ADDRESS 11 = ALWAYS ONE
                                ;                  10 +   '       '   10 = ALWAYS ONE
                                ;                  09 +   '       '   09 = ALWAYS ONE
                                ;                  08 +   '       '   08 = ALWAYS ONE
                                ;                --------
                                ;                  07 +   '       '   07  (  128)
                                ;                  06 +   '       '   06  (   64)
                                ;                  05 +   '       '   05  (   32)
                                ;                  04 +   '       '   04  (   16)
                                ;                --------
                                ;                  03 +   '       '   03  (    8)
                                ;                  02 +   '       '   02  (    4)
                                ;                  01 +   '       '   01  (    2)
                                ;                  00 +   '       '   00  (    1)
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                  UPROC   DPROC
                                ;
                                ;BUF     := 03          ; RAM DATA BUFFER                            R/W     R/W
                                ;
                                ;       DATA TO OR FROM INTERNAL REGISTERS MUST BE MOVED WITH A LOGICAL
                                ;       OPERATION.
                                ;
                                ;            15 +BUFFER  DATA   15  (32768)
                                ;            14 +   '      '    14  (16384)
                                ;            13 +   '      '    13  ( 8196)
                                ;            12 +   '      '    12  ( 4096)
                                ;          --------
                                ;            11 +   '      '    11  ( 2048)
                                ;            10 +   '      '    10  ( 1024)
                                ;            09 +   '      '    09  (  512)
                                ;            08 +   '      '    08  (  256)
                                ;          --------
                                ;            07 +   '      '    07  (  128)
                                ;            06 +   '      '    06  (   64)
                                ;            05 +   '      '    05  (   32)
                                ;            04 +   '      '    04  (   16)
                                ;          --------
                                ;            03 +   '      '    03  (    8)
                                ;            02 +   '      '    02  (    4)
                                ;            01 +   '      '    01  (    2)
                                ;            00 +   '      '    00  (    1)
```

```
                                                                        UPROC   DPROC
;UDS     := 15    ; BUS ADDR/DATA REGISTER SOURCE                       READ    (RD)
;UDD     := 05    ; BUS ADDR/DATA REGISTER DESTINATION                  WRITE   (RD)
;
;       SEE I/O REGISTER FOR FORMAT DURING INITIALIZATION
;
;       DMA TRANSFERS - DMA REQUEST @SDMA MUST BE MADE AT LEAST
;       ONE INSTRUCTION BEFORE READING OR WRITING FROM THE BUS DATA REG.
;       ONCE BUS MASTER HAS BEEN GRANTED, DMA REQUEST MAY BE DROPPED @RDMA.
;
;       THE FOLLOWING BIT DEFINITIONS APPLY TO THIS REGISTER BEING USED AS A
;       BUS ADDRESS REGISTER.
;
;       LO BUS ADDRESS BITS 0-15
;
;            15 +  BUS ADDRESS  15  (32768)
;            14 +    '        '  14  (16384)
;            13 +    '        '  13  ( 8196)
;            12 +    '        '  12  ( 4096)
;          --------
;            11 +    '        '  11  ( 2048)
;            10 +    '        '  10  ( 1024)
;            09 +    '        '  09  (  512)
;            08 +    '        '  08  (  256)
;          --------
;            07 +    '        '  07  (  128)
;            06 +    '        '  06  (   64)
;            05 +    '        '  05  (   32)
;            04 +    '        '  04  (   16)
;          --------
;            03 +    '        '  03  (    8)
;            02 +    '        '  02  (    4)
;            01 +    '        '  01  (    2)
;            00 +    '        '  00  (    1)
;
```



```
;
;       THE FOLLOWING BIT DEFINITIONS APPLY TO THIS REGISTER BEING USED AS A
;       DATA REGISTER.
;
;            15 +  BUS   DATA   15  (32768)
;            14 +    '       '  14  (16384)
;            13 +    '       '  13  ( 8196)
;            12 +    '       '  12  ( 4096)
;          --------
;            11 +    '       '  11  ( 2048)
;            10 +    '       '  10  ( 1024)
;            09 +    '       '  09  (  512)
;            08 +    '       '  08  (  256)
;          --------
;            07 +    '       '  07  (  128)
;            06 +    '       '  06  (   64)
;            05 +    '       '  05  (   32)
;            04 +    '       '  04  (   16)
;          --------
;            03 +    '       '  03  (    8)
;            02 +    '       '  02  (    4)
;            01 +    '       '  01  (    2)
;            00 +    '       '  00  (    1)
;
```

```
                                ;                                                                   UPROC   DPROC
                                ;BREG    := 06    ; BI REGISTER FOR BI CONTROL                      WRITE    -
                                ;
000001                          B.NABO   := BIT00                 ; ABORT BI COMMAND=0, NOT ABORT=1
                                ;        := BIT01                 ; NOT USED
                                ;        := BIT02                 ; NOT USED
000010                          B.BAD    := BIT03                 ; NODE NOT BAD=0, NODE BAD=1
                                ;          --------
000020                          B.LED    := BIT04                 ; YELLOW LED OFF=0, YELLOW LED ON=1
000040                          B.ODD    := BIT05                 ; BYTE BOUNDARIES IN BCAI; EVEN=0, ODD=1
000100                          B.LOOP   := BIT06                 ; BI BUS REQUEST; NORMAL MODE=0, LOOPBACK MODE=1
000200                          B.NRTY   := BIT07                 ; RETRY COMMAND; RETRY=0, NO RETRY=1
                                ;          --------
                                ;        := BIT08                 ; NOT USED
                                ;        := BIT09                 ; NOT USED
                                ;        := BIT10                 ; NOT USED
                                ;        := BIT11                 ; NOT USED
                                ;          --------
                                ;        := BIT12                 ; NOT USED
                                ;        := BIT13                 ; NOT USED
                                ;        := BIT14                 ; NOT USED
                                ;        := BIT15                 ; NOT USED
                                ;
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                 UPROC   DPROC
                                ;
                                ;SD      := 06          ; SERDES 16 DATA REGISTER                    -      R/W
                                ;
                                ;       -+--------HEADER---------+----------+--------SS-------------+---SS---+-
                                ;        !LBN!LBN!LBN!LBN!LBN!LBN!          !SYN!    DATA       !EDC!  ECC   !
                                ;       -+-----------------------+----------+--------SS-------------+---SS---+-
                                ;
                                ;       SW1 = SOFTWARE WORD 1
                                ;       SW2 = SOFTWARE WORD 2
                                ;       EDC = ERROR DETECTION CODE
                                ;       ECC = ERROR CORRECTION CODE
                                ;       SYN = 23274Q, 26BCH, 9916
                                ;
                                ;            15 +SERDES  DATA   15  (32768)
                                ;               14 +    '       '   14  (16384)
                                ;               13 +    '       '   13  ( 8196)
                                ;               12 +    '       '   12  ( 4096)
                                ;             --------
                                ;               11 +    '       '   11  ( 2048)
                                ;               10 +    '       '   10  ( 1024)
                                ;               09 +    '       '   09  (  512)
                                ;               08 +    '       '   08  (  256)
                                ;             --------
                                ;               07 +    '       '   07  (  128)
                                ;               06 +    '       '   06  (   64)
                                ;               05 +    '       '   05  (   32)
                                ;               04 +    '       '   04  (   16)
                                ;             --------
                                ;               03 +    '       '   03  (    8)
                                ;               02 +    '       '   02  (    4)
                                ;               01 +    '       '   01  (    2)
                                ;               00 +    '       '   00  (    1)
```

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;ECC     := 07          ; RS GENERATOR REGISTER (RESIDUE)              -      READ
                                ;
                                ;        := BIT15       ; 0= ECC ERROR, 1= NO ECC ERROR
                                ;        := BIT14       ; +ECC FEEDBACK ENABLE
                                ;        := BIT13       ; -WRITE ECC=NOT (ECC ENABLE AND ECC INPUT ENABLE)
                                ;        := BIT12       ; -DATA OUT
                                ;          --------
                                ;        := BIT11       ; +SERDES ENABLE
                                ;        := BIT10       ; +DTEST                                  @IOC.D
                                ;        := BIT09       ; +  RS GENERATOR 09  (  512)
                                ;        := BIT08       ; +    '        '    08  (  256)
                                ;          --------
                                ;        := BIT07       ; +    '        '    07  (  128)
                                ;        := BIT06       ; +    '        '    06  (   64)
                                ;        := BIT05       ; +    '        '    05  (   32)
                                ;        := BIT04       ; +    '        '    04  (   16)
                                ;          --------
                                ;        := BIT03       ; +    '        '    03  (    8)
                                ;        := BIT02       ; +    '        '    02  (    4)
                                ;        := BIT01       ; +    '        '    01  (    2)
                                ;        := BIT00       ; +    '        '    00  (    1)
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;BAR     := 07          ; BUFFER ADDRESS REGISTER  (BUS PROC)        WRITE     -
                                ;BAR     := 07          ; BUFFER ADDRESS REGISTER  (DRIVE PROC)        -      WRITE
                                ;
                                ;       THE LOWER 4 BITS ARE INCREMENTED EVERY TIME THE DATA BUFFER IS WRITTEN
                                ;       INTO. THEY ARE NOT INCREMENTED WHEN THE DATA BUFFER IS READ.
                                ;
                                ;            15         ; NOT IMPLEMENTED
                                ;            14         ;  "     "
                                ;            13         ; BUFFER ADDRESS 13   ( 8192)
                                ;            12         ;  "       "     12   ( 4096)
                                ;          --------
                                ;            11         ;  "       "     11   ( 2048)
                                ;            10         ;  "       "     10   ( 1024)
                                ;            09         ;  "       "     09   (  512)
                                ;            08         ;  "       "     08   (  256)
                                ;          --------
                                ;            07         ;  "       "     07   (  128)
                                ;            06         ;  "       "     06   (   64)
                                ;            05         ;  "       "     05   (   32)
                                ;            04         ;  "       "     04   (   16)
                                ;          --------
                                ;            03         ;  "       "     03   (    8)
                                ;            02         ;  "       "     02   (    4)
                                ;            01         ;  "       "     01   (    2)
                                ;            00         ;  "       "     00   (    1)
```

```
                        ;       *-------------------------------------------------------------*
                        ;       *                       BUS  I/O REGISTERS                    *
                        ;       *-------------------------------------------------------------*
                        ;
                        ;   REFERENCE "BUS DISK ADAPTER FUNCTIONAL SPECIFICATION" SECTION 4.3.4.3
                        ;
                        ;       TO SUMMARIZE THE BUS I/O REGISTER ADDRESSES AND INTERRUPTS:
                        ;
                        ;xDAIP  = 7XXXXOR.INIT= UNDEFINED              XXO =   MESSAGE RECEIVED
                        ;xDAIP  =       OR.RUN = POLL I/O BUFFER                ADAPTOR READY
                        ;xDAIP  =       OW.R/I = CONTROLLER INIT                REQUEST VAX PURGE
                        ;xDASA  = 7XXXX2W.INIT= INITIALIZATION STATUS
                        ;xDASA  = 7XXXX2R.INIT= INITIALIZATION STATUS
                        ;xDASA  = 7XXXX2R.RUN = RESERVED
                        ;xDASA  = 7XXXX2W.RUN = VAX PURGE COMPLETE (HOST RESPONSE)
                        ;
                        ;       1. ANYTIME A xDA BUS I/O PAGE ADDRESS IS SELECTED, A SLAVE SYNC
                        ;          (SSYNC L) IS RETURNED.
                        ;       2. THE xDA HAS TWO (2) MODES OF OPERATION:
                        ;             1. INITIALIZATION MODE - THE xDA IS RUNNING DIAGNOSTICS AND
                        ;                GETTING INFORMATION TO BE USED IN RUN MODE.
                        ;             2. RUN MODE - THE xDA WILL ACCEPT AND EXECUTE DCP PACKETS.
                        ;       3. INFORMATION SENT OR RECEIVED FROM THE xDA I/O PAGE IS VALID ONLY
                        ;          WHEN IN THE SPECIFIED MODE.
                        ;
                        ; *-----------------------------------------------------------------------*
                        ; *     BUS HARDWARE USAGE                                                *
                        ; *-----------------------------------------------------------------------*
                        ;
                        ;                          +-----------------------------------+
                        ;                    ON=  1|  1  2  3  4  5  6  7  8  9 10  |
                        ;        +A2 -A2           |        BUS ADDRESS SWITCH         |
                        ;       0---0---0    OFF= 0|  4  8 16 32 64 128 256 512 1K 2K  |
                        ;         W5  W4           +-----------------------------------+
                        ;
```



```
                        ;
                        ; *-----------------------------------------------------------------------*
                        ; *     xDASA INITIALIZATION REGISTER BIT DEFINITIONS                      *
                        ; *-----------------------------------------------------------------------*
                        ;
                        ; *-----------------------------------------------------------------------*
                        ; *     xDA WRITE FORMAT                                                  *
                        ; *-----------------------------------------------------------------------*
                        ;
                        ;xDASA  1W.INIT  STEP 1-3 WRITE FORMAT
                        ;
                        ;         --------
                        ;       := BIT00        ; +ERROR CODE BIT 0
                        ;       := BIT01        ; +ERROR CODE BIT 1
                        ;       := BIT02        ; +ERROR CODE BIT 2
                        ;       := BIT03        ; +ERROR CODE BIT 3
                        ;         --------
                        ;       := BIT04        ; +ERROR CODE BIT 4
                        ;       := BIT05        ; +ERROR CODE BIT 5
                        ;       := BIT06        ; +ERROR CODE BIT 6
                        ;       := BIT07        ; +ERROR CODE BIT 7
                        ;         --------
004000                  STEP1   := BIT11        ; INIT STEP 1 INDICATOR
010000                  STEP2   := BIT12        ; INIT STEP 2 INDICATOR
020000                  STEP3   := BIT13        ; INIT STEP 3 INDICATOR
040000                  STEP4   := BIT14        ; INIT STEP 4 INDICATOR
100000                  ERR     := BIT15        ; +INITIALIZATION ERROR FLAG
                        ;         --------
000200                  INTI    := BIT07        ; INIT INTERRUPT ENABLE BIT (ECHOED TO HOST)
000200                  IE      := BIT07        ; INIT INTERRUPT ENABLE BIT (ECHOED TO HOST)
100000                  OWN     := BIT15        ; 0 = HOST OWNS RING ENTRY, 1 = xDA OWNS ENTRY
040000                  FLAG    := BIT14        ; 0 = NO INTERRUPTS, 1 = INTERRUPT IF APPROPRIATE
                        ;         --------
```

```
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *      xDA READ FORMATS                                                   *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *      STEP 1 FORMAT                                                      *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;XXXX    1R.INIT  STEP 1 READ FORMAT
                                ;   BITS 15 - 0 ; +LO ORDER 16 BITS OF CSR BASE ADDRESS
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *      STEP 2 FORMAT                                                      *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;XXXX    1/3R.INIT  STEP 2 READ FORMAT
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *      STEP 3 FORMAT                                                      *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;XXXX    1R.INIT  STEP 3 READ FORMAT
                                ;
                                ;   BITS 15 - 1                 ; RESERVED
000001                          GO      := BIT00                ; +GO BIT (SET BY HOST TO PUT xDA ONLINE)
                                ;
000265                          INSTR   := <165.*110.>/100.     ;NSEC U-INSTRUCTION CYCLE TIME + DEVIATION
000552                          CYCLE   := <INSTR*2>            ;NSEC U-INSTRUCTION CYCLE TIME NOMINAL (PROCS)
```



```
                                ;
                                ; *-------------------------------------------------------------------------*
                                ; *                      LED CODE DEFINITION                                *
                                ; *-------------------------------------------------------------------------*
                                ;
                                ;                       LED LED LED LED
                                ;                        8   4   2   1
                                ;                       --- --- --- ---
                                ;                   0 =  0   0   0   0  NO ERROR (OR STUCK IN DIAGNOSTIC IF ALWAYS OFF)
                                ;                   1 =  0   0   0   1  NOT USED
                                ;                   2 =  0   0   1   0  INITIALIZATION STEP 2
                                ;                   3 =  0   0   1   1  INITIALIZATION STEP 3
                                ;                   4 =  0   1   0   0  INITIALIZATION STEP 4
                                ;                   5 =  0   1   0   1  1 SECOND BLINK FOR BUSY- NO BLINK FOR HANG
                                ;                   6 =  0   1   1   0  NOT USED
                                ;                   7 =  0   1   1   1  NOT USED
                                ;
                                ;                       ERROR CODES
                                ;                       -----------
                                ;                   8 =  1   0   0   0  INITIALIZATION (ERROR)/STEP 1 DIAGNOSTIC WRAP
                                ;                   9 =  1   0   0   1  ERROR= DIAGNOSTIC ERROR xDA #1
                                ;                   A =  1   0   1   0  ERROR= DIAGNOSTIC ERROR xDA #2
                                ;                   B =  1   0   1   1  ERROR= NOT USED
                                ;                   C =  1   1   0   0  ERROR= ERROR VECTOR = ROM  PARITY ERROR
                                ;                   D =  1   1   0   1  ERROR= ERROR VECTOR = RAM  PARITY ERROR
                                ;                   E =  1   1   1   0  ERROR= ERROR VECTOR = ROM/RAM PARITY ERROR
                                ;                   F =  1   1   1   1  ERROR= SEQUENCER ERROR
                                ;
000017                          ERREG   := R17
                                ;
                                ; ***   LED4 ON INDICATES THE xDA IS OPERATIONAL ***
                                ;
                                ;;xDAOK := LED4                          ; BIC CODE FOR xDA OK LED DISPLAY
                                ;
                                ; ***   LED1 ON INDICATES THAT THE D.PROC IS BUSY, OFF MEANS IT IS IDLE ***
                                ;
                                ;;DBUSY := LED1                          ; CODE FOR D.PROC BUSY LED DISPLAY
                                ;
                                ; ***   LED2 ON INDICATES THAT THE U.PROC IS BUSY, OFF MEANS IT IS IDLE ***
                                ;
                                ;;UBUSY := LED2                          ; CODE FOR U.PROC BUSY LED DISPLAY
```

```
                                ; *-------------------------------------------------------------------------------*
                                ; *                xDA ERROR CODES                                                *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ;       THE ERROR DISPLAY IS BIT ENCODED TO CORRESPOND TO THE ERROR MESSAGE
                                ;
                                ;               BIT 15  0 = ERROR, 1 = NO ERROR
                                ;               BIT 14  ERROR STATUS 7 BIT
                                ;               BIT 13  ERROR STATUS 6 BIT
                                ;               BIT 12  ERROR STATUS 5 BIT
                                ;
                                ;                       SELF TEST ERROR CODES
                                ;
104000                          ERR00   := <ERR+STEP1>  ;ERROR IN STACK TEST
000040                          ERR01   := 1 *32.       ;ERROR IN ALU OPERATIONAL TEST/CONTROL PROM PE/ALG&LOG PROM/CR TEST
000100                          ERR02   := 2 *32.       ;ERROR IN RAM PE/RAM BUFFER TEST
000140                          ERR03   := 3 *32.       ;ERROR in SDI/SERDES/ECC TEST
000200                          ERR04   := 4 *32.       ;ERROR in STEP 1 INIT ERROR
000240                          ERR05   := 5 *32.       ;ERROR in NPR R/W TEST
000300                          ERR06   := 6 *32.       ;ERROR in STEP 2,3, OR 4 INIT ERROR
000340                          ERR07   := 7 *32.       ;ERROR in Q BUS MEMORY TEST (MANUFACTURING ONLY)
000340                          ERR07   := 7 *32.       ;ERROR in BI TEST ERRORS
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *     HI WORD BIT DEFINTIONS FOR BDA                                            *
                                ; *-------------------------------------------------------------------------------*

100000                          BIT31   := BIT15        ; BIT 15 OF HI WORD
040000                          BIT30   := BIT14        ; BIT 14        "
020000                          BIT29   := BIT13        ; BIT 13        "
010000                          BIT28   := BIT12        ; BIT 12        "
                                ;         -------
004000                          BIT27   := BIT11        ; BIT 11        "
002000                          BIT26   := BIT10        ; BIT 10        "
001000                          BIT25   := BIT09        ; BIT 09        "
000400                          BIT24   := BIT08        ; BIT 08        "
                                ;       --------
000200                          BIT23   := BIT07        ; BIT 07        "
000100                          BIT22   := BIT06        ; BIT 06        "
000040                          BIT21   := BIT05        ; BIT 05        "
000020                          BIT20   := BIT04        ; BIT 04        "
                                ;       ---------
000010                          BIT19   := BIT03        ; BIT 03        "
000004                          BIT18   := BIT02        ; BIT 02        "
000002                          BIT17   := BIT01        ; BIT 01        "
000001                          BIT16   := BIT00        ; BIT 00        "
```

```
                                ; *-------------------------------------------------------------------------------*
                                ; *      BDA COMMAND CODES EQUATES                                                *
                                ; *-------------------------------------------------------------------------------*

000001                          BIRDCM  := 1
000001                          READ    := 1
000002                          IRCI    := 2
000003                          RCI     := 3
000004                          BIWRCM  := 4
000004                          WRITE   := 4
000005                          WCI     := 5
000006                          UWMCI   := 6
000007                          WMCI    := 7
000010                          INTR    := ^H08
000011                          IDENT   := ^H09
                                ;STOPC  := ^H0A
                                ;BISTCM := ^H0B
000014                          INVAL   := ^H0C
000015                          BDCST   := ^H0D
000016                          IPINTR  := ^H0E
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC GENERAL REGISTER OFFSET EQUATES                                     *
                                ; *-------------------------------------------------------------------------------*

000000                          BIDTYP  := ^H000        ;OFFSET TO DEVICE REGISTER
000004                          BICSR   := ^H004        ;OFFSET TO CONTROL AND STATUS REGISTER
000010                          BIBER   := ^H008        ;OFFSET TO BUS ERROR REGISTER
000014                          BIECSR  := ^H00C        ;OFFSET TO ERROR INTERRUPT CONTROL REGISTER
000020                          BIIDES  := ^H010        ;OFFSET TO INTERRUPT DESTINATION REGISTER

                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC SPECIFIC DEVICE REGISTER OFFSET EQUATES                             *
                                ; *-------------------------------------------------------------------------------*

000024                          BIIMSK  := ^H014        ;OFFSET TO IPINTR MASK REGISTER
000030                          BIPSDE  := ^H018        ;OFFSET TO FORCE IPINTR/STOP DESTINATION REGISTER
000034                          BIPSRC  := ^H01C        ;OFFSET TO IPINTR SOURCE REGISTER
000040                          BISADR  := ^H020        ;OFFSET TO STARTING ADDRESS REGISTER
000044                          BIEADR  := ^H024        ;OFFSET TO ENDING ADDRESS REGISTER
000050                          BCICSR  := ^H028        ;OFFSET TO BCI CONTROL REGISTER
000054                          BIWSTA  := ^H02C        ;OFFSET TO WRITE STATUS REGISTER
000060                          BIPSFC  := ^H030        ;OFFSET TO FORCE IPINTR/STOP COMMAND REGISTER
000100                          BIUCSR  := ^H040        ;OFFSET TO USER INTERFACE INTERRUPT CONTROL REGISTER
000360                          BIGPR0  := ^HOF0        ;OFFSET TO GENERAL PURPOSE REGISTER 0
000364                          BIGPR1  := ^HOF4        ;OFFSET TO GENERAL PURPOSE REGISTER 1
000370                          BIGPR2  := ^HOF8        ;OFFSET TO GENERAL PURPOSE REGISTER 2
000374                          BIGPR3  := ^HOFC        ;OFFSET TO GENERAL PURPOSE REGISTER 3

                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC UQSSP REGISTER OFFSET EQUATES                                       *
                                ; *-------------------------------------------------------------------------------*

000362                          IPREG   := ^HOF2        ;OFFSET TO IP REGISTER (ODD WORD BOUNDRY OF GPR 0)
000364                          SAREG   := ^HOF4        ;OFFSET TO SA REGISTER (EVEN WORD BOUNDRY OF GPR 1)
```

```
                        ; *-------------------------------------------------------------------------------*
                        ; * BIDTYP      ;^H000  ;BIIC DEVICE REGISTER                                    *
                        ; *-------------------------------------------------------------------------------*

000416                  B.DTYP  := ^H10E        ;DEVICE TYPE

                        ; *-------------------------------------------------------------------------------*
                        ; * BICSR       ;^H004  ;BIIC CONTROL AND STATUS REGISTER EQUATES                *
                        ; *-------------------------------------------------------------------------------*

100000                  HES     := BIT15        ;HARD ERROR SUMMARY
040000                  SES     := BIT14        ;SOFT ERROR SUMMARY
010000                  BROKE   := BIT12        ;BROKE BIT
004000                  STS     := BIT11        ;SELF-TEST STATUS BIT
002000                  SST     := BIT10        ;START SELF-TEST BIT

000017                  NODEID  := 17           ;NODE ID MASK

                        ; *-------------------------------------------------------------------------------*
                        ; * BIBER       ;^H008  ;BUS ERROR REGISTER                                      *
                        ; *-------------------------------------------------------------------------------*

040000                  B.NMR   := BIT30        ;NO ACK TO MULTI-RESPONDER CMD RECV
020000                  B.MTCE  := BIT29        ;MASTER TRANSMIT CHECK ERROR
010000                  B.CTE   := BIT28        ;CONTROL TRANSMIT ERROR
004000                  B.MPE   := BIT27        ;MASTER PARITY ERROR
002000                  B.ISE   := BIT26        ;INTERLOCK SEQUENCE ERROR
001000                  B.TDF   := BIT25        ;TRANSMITTER DURING FAULT
000400                  B.IVE   := BIT24        ;IDENT VECTOR ERROR
000200                  B.CPE   := BIT23        ;COMMAND PARITY ERROR
000100                  B.SPE   := BIT22        ;SLAVE PARITY ERROR
000040                  B.RDS   := BIT21        ;READ DATA SUBSTITUTE
000020                  B.RTO   := BIT20        ;RETRY TIMEOUT
000010                  B.STO   := BIT19        ;STALL TIMEOUT
000004                  B.BTO   := BIT18        ;BUS TIMEOUT
000002                  B.NEX   := BIT17        ;NON-EXISTENT ADDRESS
000001                  B.ICE   := BIT16        ;ILLEGAL CONFIRMATION ERROR

                        ; *-------------------------------------------------------------------------------*
                        ; * BISADR      ;^H020  ;STARTING ADDRESS REGISTER                               *
                        ; * BIEADR      ;^H024  ;ENDING ADDRESS REGISTER                                 *
                        ; *-------------------------------------------------------------------------------*

020000                  BA29    := BIT29        ; BUS ADDRESS 29
010000                  BA28    := BIT28        ; BUS ADDRESS 28
004000                  BA27    := BIT27        ; BUS ADDRESS 27
002000                  BA26    := BIT26        ; BUS ADDRESS 26
001000                  BA25    := BIT25        ; BUS ADDRESS 25
000400                  BA24    := BIT24        ; BUS ADDRESS 24
000200                  BA23    := BIT23        ; BUS ADDRESS 23
000100                  BA22    := BIT22        ; BUS ADDRESS 22
000040                  BA21    := BIT21        ; BUS ADDRESS 21
000020                  BA20    := BIT20        ; BUS ADDRESS 20
000010                  BA19    := BIT19        ; BUS ADDRESS 19

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```



```
000004                  BA18    := BIT18        ; BUS ADDRESS 18
000002                  BA17    := BIT17        ; BUS ADDRESS 17
000001                  BA16    := BIT16        ; BUS ADDRESS 16

037777                  BA      := BA16+BA17+BA18+BA19+BA20+BA21+BA22+BA23+BA24+BA25+BA26+BA27+BA28+BA29

                        ; *-------------------------------------------------------------------------------*
                        ; * BCICSR      ;^H028  ;BCI CONTROL REGISTER EQUATES                            *
                        ; *-------------------------------------------------------------------------------*

000001                  FORCE   := BIT16        ;FORCE COMMAND

020000                  STOPEN  := BIT13        ;STOP ENABLE
000400                  UCSREN  := BIT08        ;USER CSR SPACE ENABLE
000200                  BICSREN := BIT07        ;BIIC CSR SPACE ENABLE
000010                  RTOEVEN := BIT03        ;RTO EV CODE ENABLE

                        ; *-------------------------------------------------------------------------------*
                        ; * BIWSTA      ;^H02C  ;WRITE STATUS REGISTER                                   *
                        ; *-------------------------------------------------------------------------------*

100000                  GPR3    := BIT31        ;GENERAL PURPOSE REGISTER 3 STATUS BIT
040000                  GPR2    := BIT30        ;GENERAL PURPOSE REGISTER 2 STATUS BIT
020000                  GPR1    := BIT29        ;GENERAL PURPOSE REGISTER 1 STATUS BIT
010000                  GPR0    := BIT28        ;GENERAL PURPOSE REGISTER 0 STATUS BIT

                        ; *-------------------------------------------------------------------------------*
                        ; * BIPSFC      ;^H030  ;FORCE IPINTR/STOP COMMAND REGISTER EQUATES              *
                        ; *-------------------------------------------------------------------------------*

140000                  STOPCM  := BIT15+BIT14  ;STOP COMMAND

                        ; *-------------------------------------------------------------------------------*
                        ; * BIUCSR      ;^H040  ;USER INTERFACE INTERRUPT CONTROL REGISTER EQUATES       *
                        ; *-------------------------------------------------------------------------------*

000001                  BRLV4   :=      BIT16   ; FORCE BITS 16-19
000002                  BRLV5   :=      BIT17
000004                  BRLV6   :=      BIT18
000010                  BRLV7   :=      BIT19

000020                  BRLV4S  :=      BIT20   ; INTRC BITS 20-23
000040                  BRLV5S  :=      BIT21
000100                  BRLV6S  :=      BIT22
000200                  BRLV7S  :=      BIT23

000400                  BRLV4R  :=      BIT24   ; INTRC BITS 24-27
001000                  BRLV5R  :=      BIT25
002000                  BRLV6R  :=      BIT26
004000                  BRLV7R  :=      BIT27

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```

LSCS FORM=QUAD

```
                        ; *-------------------------------------------------------------------------*
                        ; *      BI COMMAND CODES                                                   *
                        ; *-------------------------------------------------------------------------*

140000                  QW      := BIT31+BIT30  ;OCTAWORD DATA LENGTH
100000                  QW      := BIT31        ;QUADWORD DATA LENGTH
040000                  LW      := BIT30        ;LONGWORD DATA LENGTH
000000                  WD      := 0            ;WORD DATA LENGTH (only works for BI registers)
020000                  IOACC   := BIT29        ;IO SPACE ACCESS
```



```
                                                                                        ;[E121]
                        .SBTTL  CONTROLLER RAM AND EQUATE DEFINITIONS

                        ; *-------------------------------------------------------------------------*
                        ; *      EQUATES FOR MSCP,SDI,BIT AND BYTE MASKS                            *
                        ; *-------------------------------------------------------------------------*

000100                  ATTCOD  := 100                  ; MASK FOR ATTENTION OP CODES
004000                  BTROTH  := BIT11                ; BETROTHED BIT IN GET STUD STATUS RESPONSE
000041                  BUFLMT  := 33.                  ; BUFFER CONTROL BLOCK ALLOCATION LIMIT
000400                  CCLASS  := 1*256.               ; MASS STOR. CONTROLLER CLASS FIELD FOR MSCP     [rae04]
000022                  CMODEL  := 18.                  ; CONTROLLER MODEL FIELD FOR MSCP
000024                  CMDLIM  := 20.                  ; MAXIMUM NUMBER OF MSCP PACKETS
170000                  CYLSTR  := 170000               ; MASK FOR STARTING HI CYLINDER
000377                  DEVCL   := 377                  ; MASK FOR MSCP DEVICE CLASS FIELD
001000                  DCLASS  := 1000                 ; DISK CLASS
000002                  DMBPC   := 2                    ; DM ODT PC SAVE AREA OFFSET FROM UN.BUF
000012                  DMODT   := 10.                  ; DM ODT LOCATION OFFSET FROM UN.BUF(START OF DM SPACE)
000011                  DMOVH   := 9.                   ; DM OVERLAY HI ADDR OFFSET FROM UN.BUF
000010                  DMOVL   := 8.                   ; DM OVERLAY LO ADDR OFFSET FROM UN.BUF
000013                  DMSTR   := 11.                  ; DM START ADDRESS OFFSET FROM UN.BUF
000007                  DSTSH   := 7                    ; DUST STATUS HI OFFSET FROM UN.BUF
000006                  DSTSL   := 6                    ; DUST STATUS LO OFFSET FROM UN.BUF
001000                  DUPVC   := 1000                 ; VIRTUAL CIRCUIT FOR DUP PROTOCOL
000105                  EDSEED  := 69.                  ; EDC SEED
000004                  ERRRTC  := 4                    ; ERROR RETRY COUNT
000000                  FCT.MD  := 0                    ; FCT MEDIA FORMAT WORD OFFSET
000002                  FCT.V1  := 2                    ; FCT VOLUME ID WORD 1 OFFSET
000003                  FCT.V2  := FCT.V1+1             ; FCT VOLUME ID WORD 2 OFFSET
177400                  FRMCOD  := 177400               ; MASK FOR SDI FRAME CODE
170000                  HDCOD   := 170000               ; MASK FOR SDI HEADER CODE
001000                  HDRTMO  := 512.                 ; HEADER TIMEOUT VALUE - EQUALS ABOUT 170 US.
177400                  HIBYT   := 177400               ; MASK FOR HI BYTE OF WORD
000360                  HINIB   := 360                  ; MASK FOR HI NIBBLE
000377                  LOBYT   := 377                  ; MASK FOR LO BYTE OF WORD
000377                  HOSTF   := LOBYT                ; MASK FOR HOST UNIT FLAGS
000060                  IOMSK   := 60                   ; MASK FOR MSCP I/O COMMANDS
000377                  LBNMSK  := LOBYT                ; MASK FOR SDI LBN
007400                  LBNSTR  := 7400                 ; MASK FOR STARTING HI LBN
000100                  LOGCOD  := 100                  ; MASK FOR LOG/ATTENTION OP CODES
000017                  LONIB   := 17                   ; MASK FOR LO NIBBLE
100000                  MAPENB  := BIT15                ; MAPPING ENABLED BIT IN BUFF DESC
000003                  MAPMSK  := BIT00+BIT01          ; MASK FOR MAPPING REGISTER LOW 2 BITS
077700                  MAPMSH  := 77700                ; MASK FOR UNUSED HIGH BITS
000077                  MAPMSC  := 77                   ; COMPLEMENT OF MAPMSH
100000                  MAPVAL  := BIT15                ; MAPPING VALID BIT IN PTE
126736                  MD512   := 126736               ; FCT 512 FORMAT CODE
000042                  MEMSZ   := 34.                  ; NUMBER OF PTE'S THAT WE CAN CACHE
000074                  MINUTE  := 60.                  ; 60 SECONDS IN A MINUTE !
000003                  ODTAF   := 3                    ; DM ODT ARITMETIC FLAGS OFFSET FROM UN.BUF
000005                  ODTCA   := 5                    ; DM ODT CONSOLE ADDRESS OFFSET FROM UN.BUF
000004                  ODTCY   := 4                    ; DM ODT CARRY FLAG OFFSET FROM UN.BUF
000777                  OFFMSK  := 777                  ; MASK FOR PAGE OFFSET
000020                  OFFSET  := 16.                  ; NUMBER OF WORDS IN DOWNLINE LOAD HEADER
```

```
100000                          ONLREC  := BIT15                ; ONLINE RECOVERY IN PROGRESS FLAG
000377                          OPCOD   := LOBYT                ; MASK FOR MSCP OP CODE
001000                          PAGESZ  := 512.                 ; SIZE OF HOST PAGE
000002                          PTELEN  := 2.                   ; LENGTH OF PTE IN WORDS
140000                          Q.STAT  := 140000               ; MASK FOR MSCP PACKET QUEUE STATUS
060000                          RBNCOD  := 60000                ; RBN HEADER CODE
000077                          RBNMSK  := 77                   ; MASK FOR RBN'S/TRACK
050000                          RBNPRM  := 50000                ; PRIMARY RBN HEADER CODE
030000                          RBN2ND  := 30000                ; SECONDARY RBN HEADER CODE
110000                          H11COD  := 110000               ; BAD HEADER CODE
010000                          RBNXOR  = RBNPRM&RBN2ND         ; RBNXOR = RBNPRM .XOR. RBN2ND
060000                          RBNXOR  = RBNPRM+RBN2ND-RBNXOR-RBNXOR
010000                          H11XOR  = H11COD&RBN2ND         ; H11XOR = H11COD .XOR. RBN2ND
120000                          H11XOR  = H11COD+RBN2ND-H11XOR-H11XOR
007400                          RBNSTR  := 7400                 ; MASK FOR STARTING HI RBN
000000                          RCT.MT  := 0                    ; RCT CODE FOR EMPTY ENTRY
020000                          RCT.PR  := 20000                ; RCT CODE FOR RBN USED AS PRIMARY REPLACEMENT
030000                          RCT.SE  := 30000                ; RCT CODE FOR RBN USED AS SECONDARY REPLACEMENT
100000                          RCT.EN  := 100000               ; RCT CODE FOR ENTRY PAST END OF RCT
000017                          SDIS    := 17                   ; MASK FOR CONTROL REGISTER SDI SELECT
004400                          SDITO   := 4400                 ; SDI INTERCONNECT MAX CLOCK TICK TIMEOUT
000400                          SECSZ   := 256.                 ; SECTOR SIZE
000010                          SEQUEN  := 10                   ; SEQUENTIAL COMMAND MASK
000400                          STDALN  := BIT08                ; STANDALONE BIT IN GET STUD STATUS RESPONSE
000274                          SYNCL   := ^HOBC                ; SYNC CHARACTER LOW
023000                          SYNCH   := ^HO2600              ; SYNC CHARACTER HIGH
177400                          UNITF   := HIBYT                ; MASK FOR UNIT FLAGS
000170                          CTMOUT  := 120.                 ; CONTROLLER TIMEOUT = 2 MINUTES
000440                          MODNUM  := 440                  ; CONTROLLER MODEL NUMBER (18. IN BITS 4-10)
000023                          CODVER  := 19.                  ; MICROCODE VERSION NUMBER=2.0(A000-A003=0)
                                                                ;(A004-A005=1)(A006=2)(A007=3)(A008=4)(A009=5)
                                                                ;(A010-A011=6)(A012=7)(A013-A014=8.)
                                                                ;(A015-A018=9.[mjt09])(A019=10.[mjt09a])
                                                                ;(A020=11.[mjt10],A021=11.[mjt11] A020 was
                                                                ;never released)(A022=12.[mjt12])
                                                                ;(A023=15.[mjt13])(13 & 14 not used because
                                                                ; of parity and just because it is 13.)
                                                                ;(A024=16.[E121]-[E124])
120000                          XBNCOD  := 120000               ; XBN SECTOR CODE
170000                          XBNSTR  := 170000               ; MASK FOR START XBN
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *     DEFINITION OF REFERENCES IN BUFFER MAP                                    *
                                ; *-------------------------------------------------------------------------------*
                                ; *     BASE LEVEL THREE                                                          *
                                ; *-------------------------------------------------------------------------------*
                                ; *     DIAGNOSTIC MACH     EGISTER S       EA                                    *
                                ; *-------------------------------------------------------------------------------*

000000                          DMREG7  := 0                    AGNOSTIC        REGISTER 7
000001                          DMREG1  := DMREG7+1             AGNOSTIC        REGISTER 1
000002                          DMREG2  := DMREG1+1             ; DIAGNOSTIC    REGISTER 2
000003                          DMREG3  := DMREG2+1             ; DIAGNOSTIC MA REGISTER 3
000004                          DMREG4  := DMREG3+1             ; DIAGNOSTIC MAC REGISTER 4
000005                          DMREG5  := DMREG4+1             ; DIAGNOSTIC MACHINE REGISTER 5
000006                          DMREG6  := DMREG5+1             ; DIAGNOSTIC MACHINE REGISTER 6
000007                          DMREGO  := DMREG6+1             ; DIAGNOSTIC MACHINE REGISTER O

                                ; *-------------------------------------------------------------------------------*
                                ; *     D.PROCESSOR HEADER COMPARE STORAGE AREA                                   *
                                ; *-------------------------------------------------------------------------------*

000010                          HEADER  := DMREGO+1     ; BUFFER STORAGE FOR SECTOR HEADER (HDRCMP)

000000                                  ASSUME  HEADER&17,LE,10
                                ; *-------------------------------------------------------------------------------*
                                ; *     DRIVE RESPONSE STORAGE AREA                                               *
                                ; *-------------------------------------------------------------------------------*

000011                          INPUT   := HEADER+1     ; DRIVE RESPONSE INPUT AREA
000007                          INPLEN  := 7            ; LENGTH OF DRIVE RESPONSE AREA (14 BYTES)

                                ; *-------------------------------------------------------------------------------*
                                ; *     HOST COMMUNICATION DATA AREA                                              *
                                ; *-------------------------------------------------------------------------------*

                                ; ***   NOTE THAT RSPLEN,RSPPTR,RSPCOF,RSPPOF,RCSRWO,RCSRW1 MUST BE KEPT IN THAT ORDER ***

000020                          RSPLEN  := HEADER+8.    ; RESPONSE RING BUFFER LENGTH
000021                          RSPPTR  := RSPLEN+1     ; HOST RESPONSE RING BUFFER POINTER
                                                        ; ADDR = RING BASE + 4*RSPPTR
000022                          RSPCOF  := RSPPTR+1     ; RESPONSE CURRENT RING OFFSET SAVE
000023                          RSPPOF  := RSPCOF+1     ; RESPONSE PREVIOUS RING OFFSET SAVE
000024                          RCSRWO  := RSPPOF+1     ; RESPONSE CSR WORD O SAVE
000025                          RCSRW1  := RCSRWO+1     ; RESPONSE CSR WORD 1 SAVE

000000                                  ASSUME  RCSRW1,EQ,<RSPLEN+<RCSRW1-RSPLEN>>

                                ; ***   NOTE THAT CMDLEN,CMDPTR,CMDCOF,CMDPOF,CCSRWO,CCSRW1 MUST BE KEPT IN THAT ORDER ***

000026                          CMDLEN  := RCSRW1+1     ; COMMAND RING BUFFER LENGTH
000027                          CMDPTR  := CMDLEN+1     ; HOST COMMAND RING BUFFER POINTER
                                                        ; ADDR = RING BASE + 4*(RSPLEN + CMDPTR)
000030                          CMDCOF  := CMDPTR+1     ; COMMAND CURRENT RING OFFSET SAVE
```

```
000031                          CMDPOF  := CMDCOF+1      ; COMMAND PREVIOUS RING OFFSET SAVE
000032                          CCSRWO  := CMDPOF+1      ; COMMAND CSR WORD 0 SAVE
000033                          CCSRW1  := CCSRWO+1      ; COMMAND CSR WORD 1 SAVE

000000                                  ASSUME  CCSRW1,EQ,<CMDLEN+<CCSRW1-CMDLEN>>
000002                                  RSPO    := 2     ; RESPONSE INTERRUPT RING BASE NEGATIVE OFFSET
000004                                  CMDO    := 4     ; COMMAND INTERRUPT RING BASE NEGATIVE OFFSET
000006                                  VAXO    := 6     ; VAX PURGE INTERRUPT RING BASE NEGATIVE OFFSET

                                ; *-------------------------------------------------------------------------*
                                ; *     MISCELLANEOUS WORD ALLOCATIONS  (NAME OF ROUTINE USED IN)           *
                                ; *-------------------------------------------------------------------------*

000034                          FAILUR  := CCSRW1+1      ; INTERNAL FAILURE CODE
000035                          ALOLMT  := FAILUR+1      ; ALLOCATION LIMIT (U.ALOC)
000036                          SEX2SK  := ALOLMT+1      ; NUMBER OF SECTORS UNTIL SEEK (U.ALOC)
000040                          CNVTV1  := <FAILUR+17>&7760; U.CNVT VARIABLE #1 (2 WORDS)
000042                          CNVTV2  := CNVTV1+2      ; U.CNVT VARIABLE #2 (2 WORDS)
000044                          CNVTV3  := CNVTV2+2      ; U.CNVT VARIABLE #3 (1 WORD)
000045                          CNVTV4  := CNVTV3+1      ; U.CNVT VARIABLE #4 (1 WORD)
000046                          TEMP1   := CNVTV4+1      ; TEMPORARY STORAGE #1
000047                          TEMP2   := TEMP1+1       ; TEMPORARY STORAGE #2
000050                          TEMP3   := TEMP2+1       ; TEMPORARY STORAGE #3
000051                          SEKPRV  := TEMP3+1       ; SEEK ORDERING PREVIOUS SAVE
000052                          SEKSAV  := SEKPRV+1      ; SEEK ORDERING SAVE AREA

                                ; *-------------------------------------------------------------------------*
                                ; *     ECC ROUTINE BUFFER STORAGE                                          *
                                ; *     NOTE THAT THESE LOCATIONS OVERLAY THE AREA STARTING AT 'SEX2SK'     *
                                ; *-------------------------------------------------------------------------*

000035                          ECC9.1  := FAILUR+1      ; FIRST ECC 9 WORD BUFFER AREA (ECCC)
000046                          ECC9.2  := ECC9.1+9.     ; SECOND ECC 9 WORD AREA (ECCC)
000057                          ECC9.3  := ECC9.2+9.     ; THIRD ECC 9 WORD AREA (ECCC)
000035                          ELPN    := ECC9.1
000046                          ELPM    := ECC9.2
000046                          EVAL    := ELPM          ; EVAL OVERLAYS ELPM
000057                          ELPP    := ECC9.3
000070                          ELPO    := ELPP+9.
000070                          ELOC    := ELPO          ; ELOC OVERLAYS ELPO
000112                          SY      := ELPO+18.
000112                          SYO     := SY
000122                          SY8     := SY+8.
000061                          M       := ELPP+2
000062                          DISM    := ELPP+3
000063                          DISN    := ELPP+4
000064                          LM      := ELPP+5

                                ; *-------------------------------------------------------------------------*
                                ; *     REVECTORING/REPLACEMENT STORAGE AREA                                *
                                ; *-------------------------------------------------------------------------*

000123                          REVSTR  := SY8+1         ; START OF REVECTORING RAM ASSIGNMENTS
000123                          RVCSAV  := REVSTR        ; SAVES ORIGINAL REQUEST'S I/O PARAMETERS (5 WORDS)
```



```
000130                          RVCBNL  := RVCSAV+5      ; HOLDS ORIGINAL LBN IN ERROR (LO)
000131                          RVCBNH  := RVCBNL+1      ; HOLDS ORIGINAL LBN IN ERROR (HI)
000132                          RVCTA   := RVCBNH+1      ; HOLDS ORIGINAL TRACK ADDRESS AND REAL TIME OP CODE
000133                          REVEND  := RVCTA+1       ; END OF REVECTORING STORAGE AREA

                                ; *-------------------------------------------------------------------------*
                                ; *     DYNAMIC BUFFER ALLOCATION STACK AREA                                *
                                ; *     BIT 0 OF REGISTER RLL CONTAINS THE BUFFER LOCK FLAG AND ALONG WITH  *
                                ; *     BITS 1 -> 8 OF RLL FORM THE ODD POINTER INTO THE FOLLOWING STACK    *
                                ; *     THE STACK CONTAINS 1 ENTRY FOR EACH BUFFER AVAILABLE PLUS 1 AND     *
                                ; *     THE LAST ENTRY CONTAINS A ZERO TO INDICATE THAT THERE ARE NO        *
                                ; *     BUFFERS LEFT. EACH ACTIVE STACK ENTRY CONTAINS THE ADDRESS OF A     *
                                ; *     DATA BUFFER.                                                        *
                                ; *-------------------------------------------------------------------------*

000133                          BUFPTR  = REVEND                 ; START OF BUFFER POINTER STACK

000000                          ASSUME  REVEND&1,EQ,1            ; *** NOTE - BIT 0 MUST BE ONE
000133                          BUFP00  := BUFPTR                ; *** BUFFER STACK NULL ENTRY (= 0)
000134                          ECOUNT  := BUFP00+1              ; SAVE NUMBER OF ECC SYMBOLS IN ERROR
000135                          BUFP01  := ECOUNT+1              ; *** DATA BUFFER POINTER 1
000136                          HDLMT   := BUFP01+1              ; SEARCH LIMIT COUNTER (HDRCMP,D.IOPR)
000137                          BUFP02  := HDLMT+1               ; *** DATA BUFFER POINTER 2
000140                          UBURST  := BUFP02+1              ; UNIBUS TRANSFER BURST SIZE (IN TWO WORD PAIRS)
000141                          BUFP03  := UBURST+1              ; *** DATA BUFFER POINTER 3
000142                          CLIMIT  := BUFP03+1              ; COMMAND LIMIT WORK WORD
000143                          BUFP04  := CLIMIT+1              ; *** DATA BUFFER POINTER 4
000144                          OPCODE  := BUFP04+1              ; OP CODE SAVE FOR U.ALOC,U.BMTY,U.BFIL
000145                          BUFP05  := OPCODE+1              ; *** DATA BUFFER POINTER 5
000146                          MRQUE   := BUFP05+1              ; MAINTENANCE READ QUE HEAD
000147                          BUFP06  := MRQUE+1               ; *** DATA BUFFER POINTER 6
000150                          MWQUE   := BUFP06+1              ; MAINTENANCE WRITE QUE HEAD
000151                          BUFP07  := MWQUE+1               ; *** DATA BUFFER POINTER 7
000152                          QUESAV  := BUFP07+1              ; QUEUE STATUS AND LINK WORD SAVE AREA
000153                          BUFP08  := QUESAV+1              ; *** DATA BUFFER POINTER 8
000154                          LOGLEN  := BUFP08+1              ; ATTENTION/ERROR LOG PACKET LENGTH
000155                          BUFP09  := LOGLEN+1              ; *** DATA BUFFER POINTER 9
000156                          NSEEKS  := BUFP09+1              ; NUMBER OF SEEKS TO BE INITIATED
000157                          BUFP10  := NSEEKS+1              ; *** DATA BUFFER POINTER 10
000160                          RNGBSL  := BUFP10+1              ; RING BUFFER BASE LO
000161                          BUFP11  := RNGBSL+1              ; *** DATA BUFFER POINTER 11
000162                          RNGBSH  := BUFP11+1              ; RING BUFFER BASE HI (14 BITS)
000163                          BUFP12  := RNGBSH+1              ; *** DATA BUFFER POINTER 12
000164                          INTVEC  := BUFP12+1              ; HOST INTERRUPT VECTOR (IF EQ 0 THEN NEVER INTERRUPT)
000165                          BUFP13  := INTVEC+1              ; *** DATA BUFFER POINTER 13
000166                          DMTEMP  := BUFP13+1              ; DMDT TEMPORARY SAVE AREA
000167                          BUFP14  := DMTEMP+1              ; *** DATA BUFFER POINTER 14
000170                          CNTFLG  := BUFP14+1              ; MSCP CONTROLLER FLAGS

                                        ; *-----------------------------------------------------------------*
                                        ; *     MSCP CONTROLLER FLAG BIT DEFINITIONS (LOWER BYTE)           *
                                        ; *-----------------------------------------------------------------*

000020                                  CF.THS  := BIT04         ; ENABLE ERROR LOG MESSAGES
```

```
000100                          CF.MSC  := BIT06          ; ENABLE MISCELLANEOUS ERROR LOG MESSAGES
000200                          CF.ATN  := BIT07          ; ENABLE ATTENTION MESSAGES
000320                          CFLAGS  := CF.THS!CF.ATN!CF.MSC ; FLAGS SUPPORTED BY THE CONTROLLER

000170                  CON.ST  := CNTFLG                 ; CONTROLLER STATUS WORD (HI BYTE),
                                                          ; MSCP HOST FLAGS (LO BYTE)
                                ; *-------------------------------------------------------------------*
                                ; *     CON.ST BIT DEFINITIONS  (UPPER BYTE)                          *
                                ; *-------------------------------------------------------------------*
                                ;       := BIT15          ; NOT USED
                                ;       := BIT14          ; NOT USED
                                ;       := BIT13          ; NOT USED
                                ;       := BIT12          ; NOT USED
                                ;       := BIT11          ; NOT USED
                                ;       := BIT10          ; NOT USED
001000                          LFAIL   := BIT09          ; 0 = NOP, 1 = SEND LAST FAILURE LOG PKT
000400                          VAX.PG  := BIT08          ; 0 = NO PURGE, 1 = VAX PURGE REQUIRED

000171                  BUFP15  := CNTFLG+1               ; *** DATA BUFFER POINTER 15
000172                  SAVR7   := BUFP15+1               ; FOR SAVING R7 IN ECC ROUTINE
000173                  BUFP16  := SAVR7+1                ; *** DATA BUFFER POINTER 16

                        ; *-----------------------------------------------------------------------*
                        ; *     TIMER/COUNTER SERDES SYNC STORAGE                                 *
                        ; *-----------------------------------------------------------------------*

000174                  TMR.MC  := BUFP16+1               ; HOST MSCP ACTIVITY TIMER
000175                  BUFP17  := TMR.MC+1               ; *** DATA BUFFER POINTER 17
000176                  TMR.BS  := BUFP17+1               ; CONTROLLER INTERNAL TIMER BASE
000177                  BUFP18  := TMR.BS+1               ; *** DATA BUFFER POINTER 18
000200                  HOSTMO  := BUFP18+1               ; HOST TIMEOUT (SECS), IF EQ 0 THEN NO TIMEOUTS EVER !
000201                  BUFP19  := HOSTMO+1               ; *** DATA BUFFER POINTER 19
000202                  SYNC    := BUFP19+1               ; SYNC PATTERN (LEVOWR,D.WDAT)
000203                  BUFP20  := SYNC+1                 ; *** DATA BUFFER POINTER 20

                        ; *-----------------------------------------------------------------------*
                        ; *     HOST MSCP PACKET RING BUFFER STORAGE AND POINTERS                 *
                        ; *-----------------------------------------------------------------------*

000204                  HSTQUE  := BUFP20+1               ; HOST MSCP RESPONSE QUEUE
                        ;       := BITS 11 - 0            ; HOST MSCP PACKET ADDRESS
000205                  BUFP21  := HSTQUE+1               ; *** DATA BUFFER POINTER 21
000206                  DTEMP1  := BUFP21+1               ; D.PROC TEMP STORAGE
000207                  BUFP22  := DTEMP1+1               ; *** DATA BUFFER POINTER 22
000210                  UTEMP1  := BUFP22+1               ; U.PROC TEMP STORAGE
000211                  BUFP23  := UTEMP1+1               ; *** DATA BUFFER POINTER 23
000212                  LV1SV1  := BUFP23+1               ; LEV1RD STATUS SAVE WORD
000213                  BUFP24  := LV1SV1+1               ; *** DATA BUFFER POINTER 24
000214                  LV1SV2  := BUFP24+1               ; LEV1RD OPCODE SAVE WORD
000215                  BUFP25  := LV1SV2+1               ; *** DATA BUFFER POINTER 25
000216                  RVCIND  := BUFP25+1               ; HOLD ENTRY INDICATOR (FOR U.RV57)
000217                  BUFP26  := RVCIND+1               ; *** DATA BUFFER POINTER 26
000220                  RVCRTY  := BUFP26+1               ; RETRY COUNT WHEN READING RCT
000221                  BUFP27  := RVCRTY+1               ; *** DATA BUFFER POINTER 27
```



```
000222                  RCTOFF  := BUFP27+1             ; STARTING SEARCH OFFSET IN CURRENT RCT BLOCK
000223                  BUFP28  := RCTOFF+1             ; *** DATA BUFFER POINTER 28
000224                  RCTBLK  := BUFP28+1             ; CURRENT RCT BLOCK (AS OFFSET FROM RCT BASE)
000225                  BUFP29  := RCTBLK+1             ; *** DATA BUFFER POINTER 29
000226                  RVCBUF  := BUFP29+1             ; POINTER TO ORIGINAL BUFFER IN ERROR
000227                  BUFP30  := RVCBUF+1             ; *** DATA BUFFER POINTER 30
000230                  RVCEBF  := BUFP30+1             ; HOLDS BUF.NL OF BUFFER IN ERROR
000231                  BUFP31  := RVCEBF+1             ; *** DATA BUFFER POINTER 31
000232                  RVCSDI  := BUFP31+1             ; CURRENT SDI PORT UNDERGOING REVECTORING (0 IF NONE)
000233                  BUFP32  := RVCSDI+1             ; *** DATA BUFFER POINTER 32
000234                  RVCVEC  := BUFP32+1             ; REVECTORING DISPATCH VECTOR = EVEN FOR DPROC, ODD FOR UPROC
000235                  BUFP33  := RVCVEC+1             ; *** DATA BUFFER POINTER 33
000236                  BTCNT   := BUFP33+1             ; BUFFER SERVICE FAIRNESS COUNTER
000237                  BUFP34  := BTCNT+1              ; *** DATA BUFFER POINTER 34
000240                  LN      := BUFP34+1             ; ECC VARIABLE
000241                  BUFP35  := LN+1                 ; *** DATA BUFFER POINTER 35
000242                  CMPBUF  := BUFP35+1             ;[ch04] POINTER TO DEDICATED COMPARISON BUFFER
000243                  BUFP36  := CMPBUF+1             ; *** DATA BUFFER POINTER 36
000244                  MAPSAV  := BUFP36+1             ;[UQA] TEMP SAVE FOR MAPPING INFORMATION
000245                  BUFP37  := MAPSAV+1             ; *** DATA BUFFER POINTER 37
000246                  MAPFLG  := BUFP37+1             ;[UQA]  FLAG FOR MAPPING
000247                  BUFP38  := MAPFLG+1             ; *** DATA BUFFER POINTER 38
000250                  RVECUP  := BUFP38+1             ;[UQA]  U PROC REVECTOR VECTOR
000251                  BUFP39  := RVECUP+1             ; *** DATA BUFFER POINTER 39
000252                  RVECDP  := BUFP39+1             ;[UQA] D PROC REVECTOR VECTOR
000253                  BUFP40  := RVECDP+1             ; *** DATA BUFFER POINTER 40
000254                  RVCFLG  := BUFP40+1             ;[UQA] REVECTOR PROC FLAG
000255                  BUFP41  := RVCFLG+1             ; *** DATA BUFFER POINTER 41
000256                  TEMP    := BUFP41+1             ;[V05] TEMPORARY STORAGE
000257                  BUFP42  := TEMP+1               ; *** DATA BUFFER POINTER 42
000260                  REPSTA  := BUFP42+1             ;[EERREC]HOLD REVECTOR STATUS
000261                  BUFP43  := REPSTA+1             ; *** DATA BUFFER POINTER 43
000262                  SAVUAR  := BUFP43+1             ; Storage for UAR
000263                  BUFP44  := SAVUAR+1             ; *** DATA BUFFER POINTER 44
000264                  MSCPLN  := BUFP44+1             ;[rae02] *** MSCP msg length
000265                  BUFP45  := MSCPLN+1             ; *** DATA BUFFER POINTER 45
000266                  SAVEDC  := BUFP45+1             ; [qda] Storage for edc for nonblock
000265                   BADRL  := BUFP45               ;CONTAINS LO WORD OF CONTROLLER BIIC BASE ADDRESS
000266                   BADRH  := BUFP45+1             ;CONTAINS HI WORD OF CONTROLLER BIIC BASE ADDRESS
000267                  BUFP46  := SAVEDC+1             ; *** DATA BUFFER POINTER 46
000270                  SAVBUF  := BUFP46+1             ; [qda] Storage for buffer ptr for nonblock
000271                  BUFP47  := SAVBUF+1             ; *** DATA BUFFER POINTER 47
000272                  SAVCNT  := BUFP47+1             ; [qda] Storage for word count for nonblock
000271                   RGDATL := BUFP47               ;CONTAINS LO WORD OF BIIC REGISTER DATA
000272                   RGDATH := BUFP47+1             ;CONTAINS HI WORD OF BIIC REGISTER DATA
000273                  BUFP48  := SAVCNT+1             ; *** DATA BUFFER POINTER 48
000274                  SAVADR  := BUFP48+1             ; [qda] Storage for host adr for nonblock
                         ;      := BUFP48               ;available for BDA uses
                         ;      := BUFP48+1             ;available for BDA uses
000275                  BUFP49  := SAVADR+1             ; *** DATA BUFFER POINTER 49
000276                  LGCKSV  := BUFP49+1             ; [rae07] Storage for Antilog/log check pointer
000277                  BUFP50  := LGCKSV+1             ; *** DATA BUFFER POINTER 50
000300                  BIRTRY  := BUFP50+1             ; [mjt08] bi retry count (4096)
000301                  BUFP51  := BUFP50+2             ; *** DATA BUFFER POINTER 51
```

```
000303                          BUFP52  := BUFPS1+2             ; *** DATA BUFFER POINTER 52

                                ; *-------------------------------------------------------------------------*
                                ; *     MSCP PACKET STORAGE AND DEFINITION AREA (MSCP V1.2 09-APR-82)       *
                                ; *-------------------------------------------------------------------------*

000024                          NPKTS   := CMDLIM               ; NUMBER OF MSCP PACKETS
000000                          VC.CMD  := 0                    ; COMMAND VIRTUAL CIRCUIT ID
000000                          VC.RSP  := 0                    ; RESPONSE AND ATTENTION VIRTUAL CIRCUIT ID
000020                          VC.LOG  := 1*16.                ; ERROR LOG VIRTUAL CIRCUIT ID
177760                          VC.DMM  := 177760               ; DIAGNOSTIC MACHINE MODE VIRTUAL CIRCUIT ID
000002                          P       := 2                    ; MSCP PKT OFFSET (1ST WORD IS QUEUE WORD)
                                                                ; 2ND WORD IS VIRTUAL CIRCUIT IDENTIFIER
000040                          AVL.LN  := 32.                  ; AVAILABLE ATTENTION MESSAGE LENGTH (BYTES)
000034                          BAD.LN  := 28.                  ; BUS ADDRESS ERROR LOG PACKET LENGTH (BYTES)
000030                          CNT.LN  := 24.                  ; CONTROLLER ERROR LOG PACKET LENGTH (BYTES)
000054                          DSK.LN  := 44.                  ; DISK ERROR LOG PACKET LENGTH (BYTES)
000014                          DUP.LN  := 12.                  ; DUPLICATE ATTENTION MESSAGE LENGTH (BYTES)
000034                          LOG.LN  := 28.                  ; ERROR LOG PACKET MAXIMUM LENGTH (28. WORDS)
000020                          MCP.IO  := 16.                  ; LENGTH FOR MSCP I/O END PACKETS
000030                          MCP.LN  := 24.                  ; MSCP PACKET LENGTH (WORDS)
000022                          MCP.RD  := 18.                  ; MAXIMUM MSCP PACKET READ LENGTH (WORDS)
000070                          SDI.LN  := 56.                  ; SDI ERROR LOG PACKET LENGTH (BYTES)
000304                          LOGPKT  := BUFP52+1             ; COMMON ERROR LOG AND ATTENTION PACKET
000342                          LOGEND  := LOGPKT+LOG.LN+P      ; END OF LOG PACKET (27 WORDS LONG)
000342                          PKTBUF  := <LOGEND+1>&177776    ; START OF MSCP PACKET BUFFERS (MAKE SURE ITS EVEN)
000342                          PKT001  := PKTBUF               ; 1ST  MSCP PACKET BUFFER
000374                          PKT002  := PKT001+MCP.LN+P      ; 2ND  MSCP PACKET BUFFER
000426                          PKT003  := PKT002+MCP.LN+P      ; 3RD  MSCP PACKET BUFFER
000460                          PKT004  := PKT003+MCP.LN+P      ; 4TH  MSCP PACKET BUFFER
000512                          PKT005  := PKT004+MCP.LN+P      ; 5TH  MSCP PACKET BUFFER
000544                          PKT006  := PKT005+MCP.LN+P      ; 6TH  MSCP PACKET BUFFER
000576                          PKT007  := PKT006+MCP.LN+P      ; 7TH  MSCP PACKET BUFFER
000630                          PKT008  := PKT007+MCP.LN+P      ; 8TH  MSCP PACKET BUFFER
000662                          PKT009  := PKT008+MCP.LN+P      ; 9TH  MSCP PACKET BUFFER
000714                          PKT010  := PKT009+MCP.LN+P      ; 10TH MSCP PACKET BUFFER
000746                          PKT011  := PKT010+MCP.LN+P      ; 11TH MSCP PACKET BUFFER
001000                          PKT012  := PKT011+MCP.LN+P      ; 12TH MSCP PACKET BUFFER
001032                          PKT013  := PKT012+MCP.LN+P      ; 13TH MSCP PACKET BUFFER
001064                          PKT014  := PKT013+MCP.LN+P      ; 14TH MSCP PACKET BUFFER
001116                          PKT015  := PKT014+MCP.LN+P      ; 15TH MSCP PACKET BUFFER
001150                          PKT016  := PKT015+MCP.LN+P      ; 16TH MSCP PACKET BUFFER
001202                          PKT017  := PKT016+MCP.LN+P      ; 17TH MSCP PACKET BUFFER
001234                          PKT018  := PKT017+MCP.LN+P      ; 18TH MSCP PACKET BUFFER
001266                          PKT019  := PKT018+MCP.LN+P      ; 19TH MSCP PACKET BUFFER
001320                          PKT020  := PKT019+MCP.LN+P      ; 20TH MSCP PACKET BUFFER
001352                          PKTEND  := PKT020+MCP.LN+P      ; END OF MSCP PACKETS BUFFER

                                ; *-------------------------------------------------------------------------*
                                ; *     MSCP PACKET QUEUE POINTERS                                          *
                                ; *-------------------------------------------------------------------------*

                                ; THE PACKET QUEUE POINTERS PRECEED THE FIRST WORD OF THE MSCP PACKET AREA
                                ; FOR EACH OF PACKETS. THERE ARE 'NPKTS' PACKETS, 'NPKTS' ARE USED TO
```



```
                                ; HOLD MSCP COMMANDS AND 1 IS DEDICATED FOR USE WITH 'ATTENTION' AND
                                ; 'ERROR LOG' PACKETS SENT FROM THE CONTROLLER. THIS DEDICATED 'LOG' PACKET
                                ; IS ALSO USED TO HOLD AN IMMEDIATE COMMAND WHEN THE HOST HAS ALL OF THE
                                ; OTHER PACKETS IN USE. THE QUEUE POINTERS CONSIST OF A NEXT PACKET POINTER
                                ; AND STATUS BITS (P.LINK) AS DESCRIBED BELOW:

                                        ; *---------------------------------------------------------------*
                                        ; *     RELATIVE REFERENCES TO FLAGS WITHIN A QUEUE POINTER        *
                                        ; *---------------------------------------------------------------*

100000                                  PSTAT   := BIT15        ; 0 = MSCP PACKET AVAILABLE, 1 = MSCP PKT IN USE
040000                                  PACTV   := BIT14        ; 0 = PACKET IN QUE, 1 = PACKET ACTIVE
                                        ;       := BITS 13 - 0  ; 0=LAST PACKET IN CHAIN, NEQ 0=POINTER TO NEXT PACKET

                                        ; *---------------------------------------------------------------*
                                        ; *     MSCP PACKET COMMAND OP CODES - MSCP VERSION 1.0 28-JAN-81  *
                                        ; *---------------------------------------------------------------*

000001                                  OP.ABO  :=  1           ; MSCP - ABORT OP CODE
000020                                  OP.ACC  := 20           ; MSCP - ACCESS OP CODE
000017                                  OP.ATT  := 17           ; CONTROLLER - ATTENTION PROCESSING OP CODE
000010                                  OP.AVL  := 10           ; MSCP - AVAILABLE OP CODE
000021                                  OP.CCD  := 21           ; MSCP - COMPARE CONTROLLER DATA OP CODE(NOP)
000040                                  OP.CMP  := 40           ; MSCP - COMPARE HOST DATA OP CODE
000013                                  OP.DAP  := 13           ; MSCP - DETERMINE ACCESS PATHS COMMAND
000022                                  OP.ERS  := 22           ; MSCP - ERASE OP CODE
000023                                  OP.FLU  := 23           ; MSCP - FLUSH CACHE OP CODE(NOP)
000002                                  OP.GCS  :=  2           ; MSCP - GET COMMAND STATUS
000016                                  OP.GST  := 16           ; CONTROLLER - GET DRIVE STATUS OP CODE
000003                                  OP.GUS  :=  3           ; MSCP - GET UNIT STATUS OP CODE
000030                                  OP.MRD  := 30           ; CONTROLLER - MAINTENANCE READ COMMAND
000031                                  OP.MWR  := 31           ; CONTROLLER - MAINTENANCE WRITE COMMAND
000011                                  OP.ONL  := 11           ; MSCP - ONLINE OP CODE
000041                                  OP.RD   := 41           ; MSCP - READ OP CODE
000024                                  OP.RPL  := 24           ; MSCP - REPLACE OP CODE
000004                                  OP.SCC  :=  4           ; MSCP - SET CONTROLLER CHARACTERISTICS OP CODE
000012                                  OP.SUC  := 12           ; MSCP - SET UNIT CHARACTERISTICS OP CODE
000042                                  OP.WR   := 42           ; MSCP - WRITE OP CODE

                                        ; *---------------------------------------------------------------*
                                        ; *     MSCP COMMAND PACKET OFFSETS                               *
                                        ; *---------------------------------------------------------------*

000000                                  P.LINK  := P-2          ; PACKET STATUS AND LINK WORD
000001                                  P.VCID  := P-1          ; VIRTUAL CIRCUIT IDENTIFIER
000000                                  UQ.CNT  := P.LINK       ; UQ PORT MSG LENGTH OVERLAID BY PACKET LINK          ;[E121]

                                        ;LOW 4 BITS OF P.VCID ARE CLEARED BY U.RECV & ARE USED AS FLAGS

000001                                  PABRT   := BIT00        ; 0 = OK, 1 = PACKET MARKED FOR ABORT
000002                                  PONER   := BIT01        ; 0 = OK, 1 = AVAIL PART OF FAILED ONLINE COMMAND
                                        ;       := BITS 3-2     ; UNUSED

000002                                  P.CRFO  := P+0          ; COMMAND REFERENCE NUMBER (LO)
```

```
000003                          P.CRF1  := P.CRFO+1     ; COMMAND REFERENCE NUMBER (HI)
000004                          P.UNIT  := P+2          ; UNIT NUMBER FIELD
000005                          P.RSO3  := P+3          ; RESERVED THIRD WORD
000004                          P.CONS  := P+2          ; DMDT CONSOLE ADDRESS (=0 FOR 777560)
000006                          P.OPCD  := P+4          ; OPCODE FIELD (LO BYTE)/HI BYTE=RESERVED
000007                          P.MOD   := P+5          ; MODIFIERS FIELD
000010                          P.RSO6  := P+6          ; RESERVED SIXTH WORD
000010                          P.BCNO  := P+6          ; BYTE COUNT (LO)
000011                          P.BCN1  := P.BCNO+1     ; BYTE COUNT (HI)
000012                          P.RSO8  := P+8.         ; RESERVED EIGHTH WORD
000012                          P.BUFO  := P+8.         ; BUFFER DESCRIPTOR (1ST WORD) LO 16 BITS OF ADDR
000013                          P.BUF1  := P.BUFO+1     ; BUFFER DESCRIPTOR (2ND WORD) HI 2 BITS OF ADDR
000014                          P.BUF2  := P.BUF1+1     ; BUFFER DESCRIPTOR (3RD WORD)
000015                          P.BUF3  := P.BUF2+1     ; BUFFER DESCRIPTOR (4TH WORD)
000016                          P.BUF4  := P.BUF3+1     ; BUFFER DESCRIPTOR (5TH WORD) SAVE LO LBN OF 1ST BAD BLOCK
000017                          P.BUF5  := P.BUF4+1     ; BUFFER DESCRIPTOR (6TH WORD) SAVE HI LBN OF 1ST BAD BLOCK
000012                          P.BUFL  := P.BUFO       ; CONTROLLER/MSCP BUFFER DESCRIPTOR LO
000013                          P.BUFH  := P.BUF1       ; CONTROLLER/MSCP BUFFER DESCRIPTOR HI
000020                          P.LBNO  := P+14.        ; LOGICAL BLOCK NUMBER (LO)
000021                          P.LBN1  := P.LBNO+1     ; LOGICAL BLOCK NUMBER (HI)
000010                          P.RFNO  := P+6          ; ABORT/GET COMMAND STATUS REF # (LO)
000011                          P.RFN1  := P.RFNO+1     ; ABORT/GET COMMAND STATUS REF # (HI)
000011                          P.UNFL  := P+7          ; ONLINE/SET UNIT CHAR UNIT FLAGS
000022                          P.SHUN  := P+16.        ; SHADOW UNIT
000022                          P.RS16  := P+16.        ; reserved field at offset 16.  ;[E122]
000023                          P.RS17  := P+17.        ; reserved field at offset 17.  ;[E122]
000012                          P.HSTO  := P+8.         ; HOST ID (LO)
000013                          P.HST1  := P.HSTO+1     ; HOST ID (HI)
000020                          P.ELGO  := P+14.        ; ERROR LOG FLAGS (LO)
000021                          P.ELG1  := P.ELGO+1     ; ERROR LOG FLAGS (HI)
000010                          P.RBNO  := P+6.         ; REPLACE RBN (LO)
000011                          P.RBN1  := P.RBNO+1     ; REPLACE RBN (HI)
000010                          P.VRSN  := P+6          ; SET CONTROLLER CHAR MSCP VERSION
000011                          P.CNTF  := P+7          ; SET CONTROLLER CHAR CONTROLLER FLAGS
000012                          P.HTMO  := P+8.         ; SET CONTROLLER CHAR HOST TIMEOUT
000020                          P.RGIO  := P+14.        ; MAINTENANCE READ/WRITE REGION ID (LO)
000021                          P.RGI1  := P.RGIO+1     ; MAINTENANCE READ/WRITE REGION ID (HI)
000022                          P.RGOO  := P+16.        ; MAINTENANCE READ/WRITE REGION OFFSET (LO)
000023                          P.RGO1  := P.RGOO+1     ; MAINTENANCE READ/WRITE REGION OFFSET (HI)
000026                          P.CYLS  := P+20.        ; GET UNIT STATUS CYLINDER SIZE

                        ; ***   THESE OFFSETS ARE TO STORE SDI I/O INFO AT THE END OF I/O PACKETS ***

000022                          S.ADRL  := P+16.        ; CURRENT HOST MEMORY TRANSFER ADDRESS (LO)
000023                          S.ADRH  := S.ADRL+1     ; CURRENT HOST MEMORY TRANSFER ADDRESS (HI)
000024                          S.CYLL  := P+18.        ; SDI LO CYLINDER #
000025                          S.CYLH  := S.CYLL+1     ; SDI HI CYLINDER #
000026                          S.GRUP  := S.CYLH+1     ; SDI GROUP NUMBER
000016                          S.LBNL  := P.BUF4       ; CURRENT HEADER (LBN) LO - OVERLAYS P.BUF4
000017                          S.LBNH  := P.BUF5       ; CURRENT HEADER (LBN) HI - OVERLAYS P.BUF5
000027                          S.TRAK  := S.GRUP+1     ; SDI TRACK NUMBER
000030                          S.SECS  := S.TRAK+1     ; SDI START SECTOR
000031                          S.SECI  := S.SECS+1     ; SDI INDEX SECTOR
000031                          S.OPFL  := S.SECI       ; OP CODE AND FLAGS FOR READ/WRITE COMPARE
```



```
                                ; *-------------------------------------------------------------------*
                                ; *     MSCP END PACKET OFFSETS                                       *
                                ; *-------------------------------------------------------------------*

000006                          P.FLGS  := P+4          ; FLAGS
000007                          P.STS   := P+5          ; END PACKET STATUS
000010                          P.MLUN  := P+6          ; MULTI-UNIT CODE
000020                          P.FBBO  := P+14.        ; FIRST BAD BLOCK (LO)
000021                          P.FBB1  := P.FBBO+1     ; FIRST BAD BLOCK (HI)
000012                          P.CMSO  := P+8.         ; COMMAND STATUS (LO)
000013                          P.CMS1  := P.CMSO+1     ; COMMAND STATUS (HI)
000014                          P.UNTO  := P+10.        ; GET UNIT STATUS UNIT IDENTIFIER (1ST WORD)
000015                          P.UNT1  := P.UNTO+1     ; GET UNIT STATUS UNIT IDENTIFIER (2ND WORD)
000016                          P.UNT2  := P.UNT1+1     ; GET UNIT STATUS UNIT IDENTIFIER (3RD WORD)
000017                          P.UNT3  := P.UNT2+1     ; GET UNIT STATUS UNIT IDENTIFIER (4TH WORD)
000023                          P.SHST  := P+17.        ; GET UNIT STATUS SHADOW STATUS
000024                          P.TRCK  := P+18.        ; GET UNIT STATUS TRACK SIZE
000025                          P.GRP   := P+19.        ; GET UNIT STATUS GROUP SIZE
000026                          P.CYL   := P+20.        ; GET UNIT STATUS CYLINDER SIZE
000027                          P.USVR  := P+21.        ; GET UNIT STATUS UNIT S/W & H/W VERSION NUMBERS
000030                          P.RCTS  := P+22.        ; GET UNIT STATUS RCT TABLE SIZE
000031                          P.RBNS  := P+23.        ; GET UNIT STATUS RBNS/TRACK
000031                          P.RCTC  := P+23.        ; GET UNIT STATUS RCT COPIES
000020                          P.MED1  := P+14.        ; 1ST WORD OF MEDIA TYPE
000021                          P.MED2  := P.MED1+1     ; 2ND WORD OF MEDIA TYPE
000024                          P.UNSO  := P+18.        ; ONLINE & SET UNIT CHAR UNIT SIZE (LO)
000025                          P.UNS1  := P.UNSO+1     ; ONLINE & SET UNIT CHAR UNIT SIZE (HI)
000026                          P.VSEO  := P+20.        ; ONLINE & SET UNIT CHAR VOL SERIAL NUMBER (LO)
000027                          P.VSE1  := P.VSEO+1     ; ONLINE & SET UNIT CHAR VOL SERIAL NUMBER (HI)
000025                          P.RS19  := P+19.        ; RESERVED NINETEENTH WORD
000012                          P.CTMO  := P+8.         ; SET CONTROLLER CHAR CONTROLLER TIMEOUT
000013                          P.CSVR  := P+9.         ; SET CONTROLLER CHAR CONTROLLER S/W & H/W VERSION #S
000014                          P.CNTO  := P+10.        ; SET CONTROLLER CHAR CONTROLLER ID WORD 1
000015                          P.CNT1  := P.CNTO+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 2
000016                          P.CNT2  := P.CNT1+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 3
000017                          P.CNT3  := P.CNT2+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 4

                                ; *-------------------------------------------------------------------*
                                ; *     END PACKET STATUS CODES                                       *
                                ; *-------------------------------------------------------------------*

000037                          ST.MSK  := 37           ; STATUS/EVENT CODE MASK
000040                          ST.SUB  := 40           ; SUB CODE MULTIPLIER
000007                          ST.SHF  := 7.           ; COMMAND OFFSET RIGHT ROTATE COUNT
000000                          ST.SUC  := 0            ; SUCCESS
000001                          ST.CMD  := 1            ; INVALID COMMAND
000002                          ST.ABO  := 2            ; COMMAND ABORTED
000003                          ST.OFL  := 3            ; UNIT OFFLINE
000004                          ST.AVL  := 4            ; UNIT AVAILABLE
000005                          ST.MFE  := 5            ; MEDIA FORMAT ERROR
000006                          ST.WPR  := 6            ; UNIT WRITE PROTECTED
000007                          ST.CMP  := 7            ; COMPARE ERROR
000010                          ST.DAT  := 10           ; DATA ERROR
```

```
000011                          ST.HST  := 11           ; HOST BUFFER ACCESS ERROR
000012                          ST.CNT  := 12           ; CONTROLLER ERROR
000013                          ST.DRV  := 13           ; DRIVE ERROR

                                ; *-------------------------------------------------------------------*
                                ; *     STATUS SUB-CODE VALUES                                        *
                                ; *-------------------------------------------------------------------*

                                ; *     SUCCESS SUB-CODE VALUES

000000                          SC.NML  := 0*ST.SUB     ; NORMAL SUB-CODE
000040                          SC.SDI  := 1*ST.SUB     ; SPIN DOWN IGNORED SUB-CODE
000100                          SC.SCN  := 2*ST.SUB     ; STILL CONNECTED SUB-CODE
000400                          SC.AOL  := 8.*ST.SUB    ; ALREADY ONLINE SUB-CODE

                                ; *     UNIT OFFLINE SUBCODE VALUES

000000                          SC.UNK  := 0*ST.SUB     ; UNKNOWN UNIT SUB-CODE
000040                          SC.NVL  := 1*ST.SUB     ; NO VOLUME MOUNTED OR DRIVE RUN/STOP SWITCH OUT
000100                          SC.IOP  := 2*ST.SUB     ; UNIT INOPERATIVE
000400                          SC.DIS  := 8.*ST.SUB    ; UNIT DISABLED BY FIELD SERVICE
000200                          SC.DUP  := 4*ST.SUB     ; UNIT IS A DUPLICATE (SUCCESS SUBCODE ALSO)

                                ; *     WRITE PROTECTED SUB-CODE VALUES

020000                          SC.WPH  := BIT13        ; HARDWARE WRITE PROTECTED
010000                          SC.WPS  := BIT12        ; SOFTWARE WRITE PROTECTED

                                ; *     DATA ERROR SUB-CODE VALUES (SC.ECC USED FOR MEDIA FORMAT ALSO

000000                          SC.FER  := 0*ST.SUB     ; FORCED ERROR
000100                          SC.HDR  := 2*ST.SUB     ; HEADER COMPARE ERROR
000140                          SC.DSY  := 3*ST.SUB     ; DATA SYNC TIMEOUT
000340                          SC.ECC  := 7*ST.SUB     ; ECC ERROR (UNCORRECTABLE)
000400                          SC.EC1  := 8.*ST.SUB    ; ONE SYMBOL ECC ERROR
000440                          SC.EC2  := 9.*ST.SUB    ; TWO SYMBOL ECC ERROR
000500                          SC.EC3  := 10.*ST.SUB   ; THREE SYMBOL ECC ERROR
000540                          SC.EC4  := 11.*ST.SUB   ; FOUR SYMBOL ECC ERROR
000600                          SC.EC5  := 12.*ST.SUB   ; FIVE SYMBOL ECC ERROR
000640                          SC.EC6  := 13.*ST.SUB   ; SIX SYMBOL ECC ERROR
000700                          SC.EC7  := 14.*ST.SUB   ; SEVEN SYMBOL ECC ERROR
000740                          SC.EC8  := 15.*ST.SUB   ; EIGHT SYMBOL ECC ERROR

                                ; *     MEDIA FORMAT ERROR SUB-CODES

000240                          SC.N12  := 5*ST.SUB     ; NOT 512 BYTE FORMAT
000300                          SC.BAD  := 6*ST.SUB     ; DISK NOT FORMATTED

                                ; *     DRIVE ERROR SUB-CODE VALUES

000040                          SC.STO  := 1*ST.SUB     ; SDI RESPONSE TIMEOUT
000100                          SC.INV  := 2*ST.SUB     ; INVALID SDI RESPONSE
000140                          SC.POS  := 3*ST.SUB     ; POSITIONER ERROR
000200                          SC.RWR  := 4*ST.SUB     ; LOST READ/WRITE READY
```



```
000240                          SC.DCL  := 5*ST.SUB     ; LOST DRIVE CLOCK OPERATION
000300                          SC.RRD  := 6.*ST.SUB    ; LOST DRIVE RECEIVER READY
000340                          SC.DDE  := 7.*ST.SUB    ; DRIVE DETECTED ERROR
000400                          SC.LVO  := 8.*ST.SUB    ; RTDS PULSE ERROR/RTDS PARITY ERROR/DATA PULSE ERROR

                                ; *     CONTROLLER ERROR SUB-CODE VALUES

000040                          SC.OVR  := 1.*ST.$UB    ; SERDES OVERRUN ERROR
000100                          SC.EDC  := 2.*ST.SUB    ; [ECO#1]EDC ERROR
000140                          SC.CNT  := 3.*ST.SUB    ; [ECO#1]INCONSISTENT CONTROLLER STATE

                                ; *     BUS ERROR SUB-CODE VALUES

000040                          SC.ODT  := 1*ST.SUB     ; ODD TRANSFER ADDRESS
000100                          SC.ODB  := 2*ST.SUB     ; ODD BYTE COUNT
000140                          SC.NOM  := 3*ST.SUB     ; UNIBUS NONEXISTANT MEMORY ERROR
000200                          SC.PAR  := 4*ST.SUB     ; UNIBUS PARITY ERROR
000240                          sc.imr  := 5*st.sub     ; INVALID MAPPING REGISTER

                                ; *-------------------------------------------------------------------*
                                ; *     MSCP ERROR LOG ATTENTION PACKET AND MESSAGE OFFSETS           *
                                ; *-------------------------------------------------------------------*

000002                          L.CRFO  := P+0          ; LO COMMAND REFERENCE NUMBER
000003                          L.CRF1  := L.CRFO+1     ; HI COMMAND REFERENCE NUMBER
000004                          L.UNIT  := P+2          ; UNIT NUMBER
000005                          L.SEQ   := P+3          ; SEQUENCE NUMBER
000006                          L.FMT   := P+4          ; FORMAT
000006                          L.FLGS  := P+4          ; FLAGS
000007                          L.EVNT  := P+5          ; EVENT CODE
000010                          L.CNTO  := P+6          ; CONTROLLER ID WORD 1
000011                          L.CNT1  := L.CNTO+1     ; CONTROLLER ID WORD 2
000012                          L.CNT2  := L.CNT1+1     ; CONTROLLER ID WORD 3
000013                          L.CNT3  := L.CNT2+1     ; CONTROLLER ID WORD 4
000014                          L.CSVR  := P+10.        ; CONTROLLER SOFTWARE REVISION (LO)
000014                          L.CHVR  := L.CSVR       ; CONTROLLER HARDWARE REVISION (HI)
000015                          L.MLUN  := P+11.        ; MULTI-UNIT CODE
000016                          L.BADO  := P+12.        ; HOST MEMORY ADDRESS LO
000017                          L.BAD1  := L.BADO+1     ; HOST MEMORY ADDRESS HI
000016                          L.UNTO  := P+12.        ; UNIT ID WORD 1
000017                          L.UNT1  := L.UNTO+1     ; UNIT ID WORD 2
000020                          L.UNT2  := L.UNT1+1     ; UNIT ID WORD 3
000021                          L.UNT3  := L.UNT2+1     ; UNIT ID WORD 4
000022                          L.USVR  := P+16.        ; UNIT SOFTWARE REVISION (LO)
000022                          L.UHVR  := L.USVR       ; UNIT HARDWARE REVISION (HI)
000023                          L.RTRY  := P+17.        ; RETRY COUNT
000024                          L.VSEO  := P+18.        ; VOLUME SERIAL NUMBER (LO)
000025                          L.VSE1  := L.VSEO+1     ; VOLUME SERIAL NUMBER (HI)
000026                          L.HDRO  := P+20.        ; HEADER (LO)
000027                          L.HDR1  := L.HDRO+1     ; HEADER (HI)
000030                          L.SDIO  := P+22.        ; SDI STATUS WORD 0
000031                          L.SDI1  := L.SDIO+1     ; SDI STATUS WORD 1
000032                          L.SDI2  := L.SDI1+1     ; SDI STATUS WORD 2
000033                          L.SDI3  := L.SDI2+1     ; SDI STATUS WORD 3
```

```
000034                          L.SDI4  := L.SDI3+1     ; SDI STATUS WORD 4
000035                          L.SDI5  := L.SDI4+1     ; SDI STATUS WORD 5

                                ; *-------------------------------------------------------------------*
                                ; *     LOG PACKET FLAGS                                              *
                                ; *-------------------------------------------------------------------*

040000                          LF.CON  := BIT06*256.   ; OPERATION CONTINUING FLAG
000400                          LF.SNR  := BIT00*256.   ; SEQUENCE NUMBER RESET FLAG
100000                          LF.SUC  := BIT07*256.   ; OPERATION SUCCESSFUL FLAG

                                ; *-------------------------------------------------------------------*
                                ; *     MSCP PACKET CONTROLLER TO HOST OP CODES                       *
                                ; *-------------------------------------------------------------------*

                                ; ***   VIRTUAL CIRCUIT 0       ***

000100                          OP.AVA  := 100          ; AVAILABLE ATTENTION MESSAGE
000101                          OP.DUP  := 101          ; DUPLICATE UNIT NUMBER ATTENTION MESSAGE
000102                          OP.ACP  := 102          ; ACCESS PATH ATTENTION MESSAGE
000200                          OP.END  := 200          ; END PKT FLAG OR UNRECOGNIZED COMMAND END

                                ; ***   VIRTUAL CIRCUIT 1       ***

000000                          FM.CNT  := 0            ; ERROR LOG FORMAT 1 (CONTROLLER ERROR)
000001                          FM.BAD  := 1            ; ERROR LOG FORMAT 2 (UNIBUS ACCESS ERROR)
000002                          FM.DSK  := 2            ; ERROR LOG FORMAT 3 (DISK TRANSFER ERROR)
000003                          FM.SDI  := 3            ; ERROR LOG FORMAT 4 (SDI ERROR)

                                ; *-------------------------------------------------------------------*
                                ; *     MSCP PACKET MODIFIER CODES                                    *
                                ; *-------------------------------------------------------------------*

100000                          MD.EXP  := BIT15        ; EXPRESS REQUEST MODIFIER
040000                          MD.CMP  := BIT14        ; COMPARE MODIFIER
020000                          MD.CSE  := BIT13        ; CLEAR SERIOUS EXCEPTION
020000                          MD.NOV  := BIT13        ; [CONTROLLER]NO SEEK OVERLAP MODIFIER
010000                          MD.ERR  := BIT12        ; FORCE ERROR MODIFIER
004000                          MD.SCH  := BIT11        ; SUPPRESS CACHING (HIGH)       [rae02]
002000                          MD.SCL  := BIT10        ; SUPPRESS CACHING (LOW)        [rae02]
001000                          MD.SEC  := BIT09        ; SUPPRESS ERROR CORRECTION MODIFIER
000400                          MD.SER  := BIT08        ; SUPPRESS ERROR RECOVERY MODIFIER
000200                          MD.SSH  := BIT07        ; SUPPRESS SHADOWING            [rae02]
000100                          MD.WBN  := BIT06        ; WRITE BACK (NON-VOLATILE)     [rae02]
000040                          MD.WBV  := BIT05        ; WRITE BACK (VOLATILE)         [rae02]
000020                          MD.SEQ  := BIT04        ; WRITE SHAD SET ONE UNIT AT A TIME [rae02]
000020                          MD.SHD  := BIT04        ; SHADOW UNIT SUPPLIED (ERROR ON CONTROLLER)
000002                          MD.ALL  := BIT01        ; ALL CLASS DRIVERS             [rae02]
000001                          MD.SPD  := BIT00        ; SPIN DOWN MODIFIER
000001                          MD.FEU  := BIT00        ; FLUSH ENTIRE UNIT             [rae02]
000002                          MD.VOL  := BIT01        ; VOLATILE ONLY                 [rae02]
000004                          MD.SAV  := BIT02        ; SUPPRESS AVAILABLE ATTENTION MSG MODIFIER
000001                          MD.NXU  := BIT00        ; NEXT UNIT MODIFIER
000001                          MD.RIP  := BIT00        ; ALLOW SELF DESTRUCTION MODIFIER
```



```
000004                          MD.SWP  := BIT02        ; SET WRITE PROTECT
000002                          MD.IMF  := BIT01        ; IGNORE MEDIA FORMAT ERROR MODIFIER
000001                          MD.PRI  := BIT00        ; PRIMARY REPLACEMENT MODIFIER
000001                          MD.FKC  := BIT00        ; Flush Encryption/Decryption Key Cache [ch10]
000040                          MD.EXC  := BIT05        ; Exclusive Access (unit) [ch10]

                                ; *-------------------------------------------------------------------*
                                ; *     END PACKET FLAGS                                              *
                                ; *-------------------------------------------------------------------*

100000                          EF.BBR  := BIT15        ; BAD BLOCK REPORTED
040000                          EF.BBU  := BIT14        ; BAD BLOCK UNREPORTED
020000                          EF.LOG  := BIT13        ; ERROR LOG GENERATED

                        ; *-------------------------------------------------------------------*
                        ; *     REFERENCES WITHIN THE SDI INTERCONNECT CONTROL BLOCK           *
                        ; *-------------------------------------------------------------------*

000000                  SDI.ST  := 0            ; STATUS OF SDI INTERCONNECT

                                ; *-------------------------------------------------------------------*
                                ; *     SDI INTERCONNECT STATUS FLAG EQUATES                          *
                                ; *-------------------------------------------------------------------*

000001                          PKIP    := BIT00        ; 0 = NO PKT, 1 = PACKET IN PROGRESS
000002                          SEEK    := BIT01        ; 0 = NO SEEK NEEDED, 1 = SEEK INITIATE NEEDED
000004                          DERR    := BIT02        ; 0 = NO ERROR, 1 = FATAL ERROR ON I/O COMMAND
000010                          VECT    := BIT03        ; 0 = VECTOR LEVEL DONE, 1 = STATE VECTOR LEVEL ACTIVE
000020                          BFRQ    := BIT04        ; 0 = OK, 1 = BUFFER CONTROL BLOCKS NEEDED
000040                          SUSP    := BIT05        ; 0 = TASK RUNNING, 1 = TRANSFER TASK SUSPENDED
000100                          BFSV    := BIT06        ; 0 = OK, 1 = BUFFER SERVICE REQUIRED FROM U.PROC
000200                          XCMP    := BIT07        ; 0 = OK, 1 = TRANSFER TASK COMPLETED
000400                          DATT    := BIT08        ; 0 = OK, 1 = DRIVE ATTENTION PENDING
001000                          GSEL    := BIT09        ; MODIFIES "SEEK"; 0=SEEK, 1=GROUP SELECT ONLY          ;[E121]
002000                          DRDUP   := BIT10        ; DRDUP, DRVOL, AND DRAVL ARE ENCODED AS FOLLOWS:
004000                          DRVOL   := BIT11        ; 000 = DRIVE ONLINE
010000                          DRAVL   := BIT12        ; 001 = DRIVE ONLINE AND DUPLICATE
                                                        ; 010 = DRIVE OFFLINE AND NOT USABLE
                                                        ; 011 = DRIVE OFFLINE AND DUPLICATE
                                                        ; 100 = DRIVE AVAILABLE (SPINABLE OR NOT SPINABLE)
                                                        ; 101 = NOT USED
                                                        ; 110 = NOT USED
                                                        ; 111 = DRIVE OFFLINE AND UNKNOWN TO CONTROLLER
020000                          SLAT    := BIT13        ; 0 = NOP, 1 = SEND LOG OR ATTENTION PKT TO HOST
                                                        ; IN ADDITION SLAT IS USED AS A FLAG TO AID IN
                                                        ; DETERMINING IF A DRIVE IS ALREADY ONLINE TO THE CONTROLLER
040000                          ERRIP   := BIT14        ; 0 = NO ERROR RECOVERY, 1 = LEVEL 0 RECOVERY ACTIVE
                                                        ; IN ADDITION ERRIP IS USED AFTER INITIALIZATION
                                                        ; TO FORCE INIT TO ALL DRIVES
100000                          RVCT    := BIT15        ; 0 = NO REVECTORING REQUIRED, 1 = REVECTORING REQUIRED
                                                        ; IF THIS BIT = 1 AND "RVCSDI" POINTS TO THIS SDI BLOCK,
                                                        ; THEN REVECTORING IS IN PROGRESS FOR THIS DRIVE.

000001                  SDI.SL  := SDI.ST+1     ; HEADER COMPARE SEARCH LIMIT
```

```
000002                          SDI.CP  := SDI.SL+1     ; POINTER TO CURRENTLY ACTIVE UNIT CHARACTERISTICS
000003                          SDI.SW  := SDI.CP+1     ; STATE WORD - MSCP END STATUS/ERROR LOG CODE
000004                          SDI.TM  := SDI.SW+1     ; SDI INTERCONNECT ACKNOWLEDGE/COMMAND TIMER
000005                          SDI.UG  := SDI.TM+1     ; U.PROC GROUP WORD (USED BY U.CPRM)
000006                          SDI.UB  := SDI.UG+1     ; ADDRESS OF U.PROC'S BUFFER CONTROL BLOCK(FOR I/O)
000007                          SDI.DB  := SDI.UB+1     ; ADDRESS OF D.PROC'S BUFFER CONTROL BLOCK(FOR I/O)

                                ; ***   NOTE - THE FOLLOWING 4 WORDS (SDI.1T,SDI.CL,SDI.CH,SDI.GP) ARE USED TO SEND
                                ; ***   THE SDI LEVEL 2 SEEK COMMAND DIRECTLY FROM THE SDI CONTROL BLOCK

000010                          SDI.1T  := SDI.DB+1     ; TEMP WORD (USED FOR ERROR LOG/ATTN CODE,SEEK COMMAND)
000011                          SDI.CL  := SDI.1T+1     ; CURRENT CYLINDER ADDRESS LO (FORMS SEEK COMMAND)
000012                          SDI.CH  := SDI.CL+1     ; CURRENT CYLINDER ADDRESS HI (FORMS SEEK COMMAND)
000013                          SDI.GP  := SDI.CH+1     ; CURRENT GROUP NUMBER  (FORMS SEEK COMMAND)
000014                          SDI.SS  := SDI.GP+1     ; # OF SECTORS TO TRANSFER BEFORE SEEK REQUIRED
000015                          SDI.XL  := SDI.SS+1     ; LO NUMBER OF BYTES REMAINING TO BE TRANSFERRED
000016                          SDI.XH  := SDI.XL+1     ; HI NUMBER OF BYTES REMAINING TO BE TRANSFERRED

                                ; ***   SDI CONTROL BLOCK ERROR STATE CONTROL WORDS ***

000017                          SDI.E1  := SDI.XH+1     ; LEVEL 1 ERROR STATE CONTROL WORD
                                                        ; USED FOR READ RETRIES (EDC AND ECC ERRORS)
000020                          SDI.E0  := SDI.E1+1     ; LEVEL 0 ERROR STATE CONTROL WORD
                                                        ; USED FOR SDI LEVEL 2,1 ERRORS
                                                        ; AND FOR READ,WRITE,COMPARE DRIVE ERROR RETRIES
000377                                  ERRVEC  := LOBYT        ; ERROR STATE VECTOR (SDI.E0 AND SDI.E1)
                                                                ; 0 = LEVEL INACTIVE, NEQ 0 = LEVEL ACTIVE
001400                                  LV2CNT  := 1400         ; SDI LEVEL TWO RETRY COUNT (2 BITS LONG)
                                                                ; INCREMENT RETRY COUNT WITH ADDC #377,REG
                                                                ; USED WITH SDI.E0
007400                                  RETCNT  := 7400         ; I/O RETRY COUNT (USED WITH SDI.E0)
036000                                  IORTY   := 36000        ; I/O RETRY INDICATOR FLAGS
002000                                          IOCNT   := BIT10        ; READ/WRITE RETRY INDICATOR (1=LAST RETRY)
004000                                          IOSEK   := BIT11        ; SEEK ERROR (RECAL,RESEEK IN RECOVERY)
010000                                          IOCLK   := BIT12        ; DRIVE CLOCK TIMEOUT ERROR (INIT,RESEEK IN RECOVERY)
020000                                          IORWR   := BIT13        ; R/W RDY OR SERDES LATE ERROR (RETRY I/O IN RECOVERY)
040000                                  ERRINI  := BIT14        ; 0 = NO INIT DONE, 1 = INIT DONE ON RECOVERY
                                                                ; USED WITH SDI.E0
100000                                  ERRINP  := BIT15        ; 0 = NO RECOVERY, 1 = RECOVERY IN PROGRESS
                                                                ; USED WITH SDI.E0 AND SDI.E1
000021                          SDI.SV  := SDI.E0+1     ; SDI LEVEL 2 FUNCTION STATE VECTOR
000022                          SDI.PQ  := SDI.SV+1     ; POINTER TO HEAD OF CONTROL BLOCK MSCP PACKET QUEUE
000023                          SDI.RO  := SDI.PQ+1     ; ROTATIONAL OPTIMIZATION COUNT
000024                          SDI.OE  := SDI.RO+1     ; SDI OVERLAP ENABLE FLAG;
                                                        ; 0 = DISABLE, 1 = ALLOW, -1 = D.PROC SAW LAST BUFFER
000025                          SDI.OM  := SDI.OE+1     ; STORAGE FOR CURRENT MSCP PKT PTR WHILE OVERLAPPING
000026                          SDI.ES  := SDI.OM+1     ; SDI EXTENDED STATUS
000001                                  RVWRIT  := BIT00        ; REVECTOR WRITE FLAG FOR U.BFLC
000002                                  OFFTRK  := BIT01        ; SEEK REQUIRED DUE TO ERROR RECOVERY LEVEL USED
000004                                  FSEEK   := BIT02        ; FORCE SEEK FLAG
                                        ;       := BIT03        ; NOT USED
000020                                  RVACTV  := BIT04        ; REVECTORING ACTIVE
000040                                  URETRY  := BIT05        ; U.PROC RETRY IN PROCESS
                                        ;       := BIT06-BIT07  ; UNUSED
```



```
                                        ;       := BIT08-BIT14  ; SEEK FAIRNESS COUNTER
077400                                  FAIRCT  := 77400        ; SEEK FAIRNESS COUNT MASK
100000                                  ELEV    := BIT15        ; SEEK ELEVATOR FLAG

000027                          SDI.BL  := SDI.ES+1     ; LO BAD BLOCK DETECTED (CLEAR IN U.ALOC)
000030                          SDI.BH  := SDI.BL+1     ; HI BAD BLOCK DETECTED (CLEAR IN U.ALOC)
000031                          SDI.RL  := SDI.BH+1     ; LO LBN OF PRIMARY RBN
000032                          SDI.RH  := SDI.RL+1     ; HI LBN OF PRIMARY RBN

                                ; ***   START OF DRIVE GENERAL CHARACTERISTICS  (SDI V3.7, 15-JUNE-81)

000033                          SDI.TO  := SDI.RH+1     ; LO BYTE = SHORT T.O. (LO)/SDI VERS (HI)
000033                          SDI.XR  := SDI.TO       ; HI BYTE = TRANSFER RATE
000034                          SDI.LT  := SDI.TO+1     ; LO BYTE = LONG T.O. (LO)/RETRY # (HI)
000034                          SDI.RC  := SDI.LT       ; HI BYTE = RCT COPIES (LO)/RESERVED (HI)
100000                                  UNC.SS  := BIT15        ; 0 = 512 BYTE ONLY, 1 = 512 OR 576 BYTE SECT
000035                          SDI.ER  := SDI.LT+1     ; LO BYTE = ERROR RECOVERY LEVELS
000035                          SDI.EC  := SDI.ER       ; HI BYTE = ECC ERROR THRESHOLD
000036                          SDI.RS  := SDI.ER+1     ; HARDWARE REVISION #(HI)/SOFTWARE REVISION #(LO)
100000                                  FDIAG   := BIT15        ; 1= LIMITED DIAG, 0 = FULL DIAG SUPPORT        ;[E121]
000037                          SDI.I1  := SDI.RS+1     ; UNIQUE DRIVE ID #1
000040                          SDI.I2  := SDI.I1+1     ; UNIQUE DRIVE ID #2
000041                          SDI.I3  := SDI.I2+1     ; UNIQUE DRIVE ID #3
000042                          SDI.TI  := SDI.I3+1     ; DRIVE TYPE IDENTIFIER (LO)/DRIVE REVS/SEC (HI)
000043                          SDI.UE  := SDI.TI+1     ; END OF CONTROLLER CRITICAL DRIVE COMMON CHARACTERISTICS
000010                          SDI.DL  := <SDI.TI+1-SDI.TO>; LENGTH OF DRIVE COMMON CHARACTERISTICS

                                ; ***   START OF SDI STATUS STORAGE (7 WORDS) ***

000043                          SDI.UN  := SDI.UE       ; SDI STATUS WORD 0 (UNIT NUMBER INFORMATION)

                                ; *-------------------------------------------------------------------*
                                ; *     DRIVE STATUS UNIT WORD BIT DEFINITIONS                        *
                                ; *-------------------------------------------------------------------*

010000                                  DRV.U1  := BIT12        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
020000                                  DRV.U2  := BIT13        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
040000                                  DRV.U3  := BIT14        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
100000                                  DRV.U4  := BIT15        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
170000                                  DRV.SU  := <DRV.U1+DRV.U2+DRV.U3+DRV.U4>; SUBUNIT MASK

000044                          SDI.S1  := SDI.UN+1     ; SDI STATUS WORD 1

                                ; *-------------------------------------------------------------------*
                                ; *     DRIVE STATUS 1ST WORD BIT DEFINITIONS                         *
                                ; *-------------------------------------------------------------------*

000001                                  DRV.RU  := BIT00        ; 0 = RUN/STOP SWIT OUT, 1 = RUN/STOP SWIT IN
000002                                  DRV.PS  := BIT01        ; 0 = PORT SWITCH OUT, 1 = PORT SWITCH IN
                                        ;       := BIT02        ; NOT USED
000010                                  DRV.EL  := BIT03        ; 0 = NO ACTION, 1 = PLEASE SEND A LOG PACKET
000020                                  DRV.SR  := BIT04        ; 0 = SPINDLE NOT AT SPEED, 1 = SPINDLE AT SPEED
000040                                  DRV.DR  := BIT05        ; 0 = NO DIAG REQ'D, 1 = DIAGNOSTIC LOAD REQ'D
000100                                  DRV.RR  := BIT06        ; 0 = NO READJ REQ'D, 1 = READJUSTMENT REQ'D
```

```
000200                          DRV.OA  := BIT07        ; 0 = ONLINE/AVAILABLE, 1 = ONLINE TO OTHER PORT
000400                          DRV.S7  := BIT08        ; 0 = 512 BYTE SECTOR, 1 = 576 BYTE SECTOR
001000                          DRV.DB  := BIT09        ; 0 = NO DIAG CYL ACCESS, 1 = YES DIAG CYL ACCESS
002000                          DRV.FO  := BIT10        ; 0 = NO FORMATTING, 1 = FORMATTING ENABLED
004000                          DRV.DD  := BIT11        ; 0 = DRIVE ENABLED, 1 = DRIVE DISABLED FOR DIAGNOSTICS
010000                          DRV.W1  := BIT12        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
020000                          DRV.W2  := BIT13        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
040000                          DRV.W3  := BIT14        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
100000                          DRV.W4  := BIT15        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
000045                          SDI.S2  := SDI.S1+1     ; SDI STATUS WORD 2

                                ; *-------------------------------------------------------------------*
                                ; * LOWER BYTE OF SDI.S2 IS STATUS ERROR BYTE                         *
                                ; *-------------------------------------------------------------------*
                                ;       := BIT00        ; NOT USED
                                ;       := BIT01        ; NOT USED
                                ;       := BIT02        ; NOT USED
000010                          DRV.WE  := BIT03        ; 0 = OK, 1 = WRITE LOCK ERROR
000020                          DRV.DF  := BIT04        ; 0 = INIT DIAG OK, 1 = INIT DIAG FAILURE
000040                          DRV.PE  := BIT05        ; 0 = OK, 1 = PROTOCOL ERROR
000100                          DRV.RE  := BIT06        ; 0 = OK, 1 = RETRYABLE ERROR-REISSUE COMMAND
000200                          DRV.DE  := BIT07        ; 0 = OK, 1 = DRIVE ERROR
000170                          DCMASK  := <DRV.WE+DRV.DF+DRV.PE+DRV.RE>
                                                        ; DRIVE CLEAR COMMAND ERROR CLEAR MASK
000400                          DRV.C4  := BIT08        ; CONTROLLER STATUS - IGNORED BY DRIVE
001000                          DRV.C3  := BIT09        ; CONTROLLER STATUS - IGNORED BY DRIVE
002000                          DRV.C2  := BIT10        ; CONTROLLER STATUS - IGNORED BY DRIVE
004000                          DRV.C1  := BIT11        ; CONTROLLER STATUS - IGNORED BY DRIVE
010000                          DRV.S1  := BIT12        ; CONTROLLER STATUS - IGNORED BY DRIVE
020000                          DRV.S2  := BIT13        ; CONTROLLER STATUS - IGNORED BY DRIVE
040000                          DRV.S3  := BIT14        ; CONTROLLER STATUS - IGNORED BY DRIVE
100000                          DRV.S4  := BIT15        ; CONTROLLER STATUS - IGNORED BY DRIVE
170000                          DRV.SN  := DRV.S1+DRV.S2+DRV.S3+DRV.S4 ; MASK FOR TESTING S BITS

                        ; ***   SDI CONTROL BLOCK PRIVATE ATTENTION PACKET ***
                        ; ***   NOTE - BIT 0 OF THE PACKET CONTROL WORD ADDRESS DIFFERENTIATES BETWEEN
                        ; ***   ATTENTION PACKETS AND NORMAL MSCP PACKETS, I.E. BIT 0 = 1 MEANS IT IS
                        ; ***   AN INTERNAL ATTENTION PACKET AND BIT 0 = 0 MEANS IT IS A MSCP PACKET.

000046                  SDI.AT  := SDI.S2+1             ; ATTENTION PACKET CONTROL WORD
000047                  SDI.2T  := SDI.AT+1             ; 2ND TEMP WORD - ATTENTION VIRTUAL CIRCUIT NOT NEEDED
000050                  SDI.S4  := SDI.AT+P.CRF0        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000051                  SDI.S5  := SDI.AT+P.CRF1        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000052                  SDI.S6  := SDI.AT+P.UNIT        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000053                  SDI.S7  := SDI.AT+P.RS03        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000054                  SDI.OP  := SDI.AT+P.OPCD        ; USED FOR ATTENTION/GET STATUS INTERNAL OP CODE
000055                  SDI.PL  := SDI.AT+P.MOD         ; WRITE PRELOAD COUNTER (NOTE MSB MUST BE ZERO)
                        ;SDIB.L := SDI.AT+8.            ; LENGTH OF SDI CONTROL BLOCK

                        ; ***   START OF SUBUNIT CHARACTERISTICS BUFFER ***

                        ;SDI.CW := 0             ; SUBUNIT BUFFER CONTROL WORD
000056                  SDI.CW  := SDI.PL+1      ; [V05]SUBUNIT BUFFER CONTROL WORD
                                ;       BITS 15-12      ; SUBUNIT CODE (0001,0010,0100,1000)
```



```
004000                          DRV.AV  := BIT11        ; 0 = SUBUNIT ONLINE, 1 = SUBUNIT AVAILABLE
002000                          DRV.AT  := BIT10        ; 0 = ATTENTION SENT, 1 = NEED TO SEND ATTENTION
                                ;       BITS 9-0        ; UNIT NUMBER (0 - 1023) FOR THIS SUBUNIT
176000                          DRV.UM  := DRV.SUI!DRV.AT!DRV.AV; UNIT NUMBER MASK (USE WITH 'BIC')
                                ; IF SDI.CW IS ZERO THEN IT IS AVAILABLE, OTHERWISE IT IS IN USE.

000057                  SDI.SD  := SDI.CW+1     ; PARENT SDI CONTROL BLOCK POINTER
000060                  SDI.UF  := SDI.SD+1     ; UNIT FLAGS FOR MSCP PACKET

                                ; *-------------------------------------------------------------------*
                                ; *     UNIT FLAG BIT DEFINITIONS                                     *
                                ; *-------------------------------------------------------------------*

100000                          UF.RPL  := BIT15        ; 1 = NO HOST BBR ON THIS UNIT                          ;[E121]
020000                          UF.WPH  := BIT13        ; 1 = WRITE PROTECTED HARDWARE
010000                          UF.WPS  := BIT12        ; 1 = WRITE PROTECTED SOFTWARE
000200                          UF.RMV  := BIT07        ; 1 = REMOVABLE MEDIA
000004                          UF.576  := BIT02        ; 1 = VOLUME MOUNTED HAS 576 BYTE SECTORS
000002                          UF.CMW  := BIT01        ; 1 = COMPARE ON ALL WRITE OPERATIONS
000001                          UF.CMR  := BIT00        ; 1 = COMPARE ON ALL READ OPERATIONS
000003                          UF.MSK  := <UF.CMW+UF.CMR>

000061                  SDI.V1  := SDI.UF+1     ; VOLUME SERIAL NUMBER WORD 1
000062                  SDI.V2  := SDI.V1+1     ; VOLUME SERIAL NUMBER WORD 2
000063                  SDI.H1  := SDI.V2+1     ; LO ORDER LBN SPACE IN CYLINDERS
000064                  SDI.H2  := SDI.H1+1     ; HI ORDER LBN SPACE IN CYLS (LO)/HI CYL # (HI)
000065                  SDI.GC  := SDI.H2+1     ; GROUPS PER CYL (LO)//HI START LBN (LO)/HI START XBN (HI)
000066                  SDI.TG  := SDI.GC+1     ; TRACKS PER GROUP (LO)//HI START RBN (LO)/HI START DBN (HI)
000067                  SDI.RT  := SDI.TG+1     ; # REPLACEMENTS PER TRACK,RM FLAG (LO)/RESERVED (HI)

                                ; *-------------------------------------------------------------------*
                                ; *     SUBUNIT CAPABILITIES BIT DEFINITIONS                          *
                                ; *-------------------------------------------------------------------*

000200                          UNC.RM  := BIT07        ; 0 = NO REMOVABLE MEDIA, 1 = REMOVABLE MEDIA
000070                  SDI.DP  := SDI.RT+1     ; DATA PREAMBLE SIZE (LO)/HEADER PREAMBLE SIZE (HI)
000071                  SDI.M1  := SDI.DP+1     ; MEDIA TYPE LO ORDER
000072                  SDI.M2  := SDI.M1+1     ; MEDIA TYPE HI ORDER
000073                  SDI.FC  := SDI.M2+1     ; FCT COPY SIZE IN XBNS
000074                  SDI.12  := SDI.FC+1     ; # 512 BYTE LBNS PER TRACK (LO)
000074                  SDI.GO  := SDI.12       ; GROUP OFFSET (HI)
000075                  SDI.L1  := SDI.GO+1     ; LBN'S IN HOST AREA (LO)
000076                  SDI.L2  := SDI.L1+1     ; LBN'S IN HOST AREA (HI)
000077                  SDI.RV  := SDI.L2+1     ; RCT SIZE IN LBN'S
000100                  SDI.PH  := SDI.RV+1     ;[V05] TEMP FOR PRIMARY HDR DURING REVECT
000101                  SDI.LL  := SDI.PH+1     ; LENGTH OF UNIT CHARACTERISTICS

                                ; *-------------------------------------------------------------------*
                                ; *     HOST MAPPING INFORMATION                                      *
                                ; *-------------------------------------------------------------------*

000101                  MAP.CH  := SDI.PH+1     ;PTR TO START OF PTE CACHE
000102                  MAP.NX  := MAP.CH+1     ;PTR TO CURRENT PTE ENTRY
000103                  MAP.MO  := MAP.NX+1     ;ADDRESS OF NEXT UNREAD PTE IN HOST MEM LOW
```

```
000104                          MAP.M1  := MAP.MO+1      ;ADDRESS OF NEXT UNREAD PTR IN HOST MEM HIGH
000105                          MAP.RD  := MAP.M1+1      ;NUMBER OF CACHED BUT UNUSED PTE'S
000106                          MAP.UR  := MAP.RD+1      ;NUMBER OF REMAINING UNREAD PTE'S FOR THIS REQ
000107                          MAP.OF  := MAP.UR+1      ;OFFSET INTO PAGE
000110                          MAP.S1  := MAP.OF+1      ;SEGMENT 1 SIZE
000111                          MAP.VO  := MAP.S1+1      ;LOW ORDER MAPPING REGISTER NUMBER
000112                          MAP.V1  := MAP.VO+1      ;HIGH ORDER MAPPING REGISTER NUMBER
000113                          MAP.V2  := MAP.V1+1      ;LOW ORDER MAPPING BASE ADDRESS
000114                          MAP.V3  := MAP.V2+1      ;HIGH ORDER MAPPING BASE ADDRESS
000115                          MAP.ST  := MAP.V3+1      ;STATUS -- 1 = MAP INIT DONE, 0 = NOT DONE [kjk]
000116                          MAP.ND  := MAP.ST+1      ;END OF MAPPING INFORMATION
000120                          SDIB.L  := MAP.ND+2.     ;[V05] LENGTH OF SDI CONTROL BLOCK PLUS EXTRA

                                ; *-------------------------------------------------------------------------*
                                ; *     DATA BUFFER CONTROL BLOCK DEFINITIONS                               *
                                ; *-------------------------------------------------------------------------*

000000                          BUF.NL  := 0             ; POINTER TO NEXT DATA BUFFER CONTROL BLOCK

                                        ; *-------------------------------------------------------------------*
                                        ; *     DATA BUFFER USE FLAGS (UPPER 2 BITS OF BUF.NL)                *
                                        ; *-------------------------------------------------------------------*

100000                                  BLAST   := BIT15         ; 0 = CONTINUE, 1 = LAST BUF/SEEK REQ
040000                                  BFULL   := BIT14         ; 0 = BUFFER EMPTY, 1 = BUFFER FULL

000001                          BUF.ST  := BUF.NL+1      ; BUFFER CONTROL BLOCK STATUS WORD

                                        ; *-------------------------------------------------------------------*
                                        ; *     BUFFER CONTROL BLOCK STATUS BIT DEFINITIONS                   *
                                        ; *-------------------------------------------------------------------*

100000                                  BRTRY   := BIT15         ; 0 = NORMAL, 1 = U.PROC READ RETRY
040000                                  BLSTB   := BIT14         ; 0 = NOT LAST, 1 = RETRY REALLY IS LAST BUFFER
020000                                  BECER   := BIT13         ; 0 = NORMAL, 1 = FATAL ECC ERROR, BUT GIVE BEST GUESS DATA
010000                                  BECC    := BIT12         ; 0 = NO ECC ERROR, 1 = ECC ERROR
004000                                  BGOOD   := BIT11         ; 0 = NO HEADERS OK, 1 = AT LEAST 1 HEADER OK
002000                                  BDSNF   := BIT10         ; 0 = NO DATA SYNC ERR, 1 = DATA SYNC NOT FOUND [EERREC]
001000                                  BLRWR   := BIT09         ; 0 = NO RD/WR RDY ERR, 1 = LOST RD/WR RDY [EERREC]
000400                                  BERDN   := BIT08         ; 0 = NO ERR REC CMD ISSUED, 1 = ERR REC CMD ISSSUED [EERREC]
                                        ;       := BITS 7-6      ; NOT USED
000040                                  BRCTS   := BIT05         ; 0 = RCT HAS NOT BEEN SEARCHED, 1 = HAS BEEN SEARCHED [EERREC]
000020                                  BRARS   := BIT04         ; 0 = RCT HAS NOT ALREADY BEEN SEARCHED, 1 = HAS ALREADY BEEN SEARCHED [EE
000010                                  BMAPDN  := BIT03         ; 0 = MAPPING NOT DONE, 1 = MAPPING DONE
000004                                  BNXCOP  := BIT02         ; 0 = FATAL ERROR, 1 = READ NEXT RCT COPY
000002                                  BCGRP   := BIT01         ; 0 = SELECT GROUP NOT DONE, 1 = SELECT GROUP DONE
000001                                  BGRUP   := BIT00         ; 0 = NO ACTION, 1 = SELECT GROUP

000002                          BUF.BP  := BUF.ST+1      ; DATA BUFFER POINTER
000003                          BUF.HL  := BUF.BP+1      ; 1ST WORD OF EXPECTED HEADER
000004                          BUF.HH  := BUF.HL+1      ; 2ND WORD OF EXPECTED HEADER
000005                          BUF.TA  := BUF.HH+1      ; TRACK ADDRESS AND I/O COMMAND THIS SECTOR
000006                          BUF.SD  := BUF.TA+1      ; POINTER TO PARENT SDI CONTROL BLOCK
000007                          BUF.UA  := BUF.SD+1      ; HOST ADDRESS BITS 15 - 0 (FOR THIS SECTOR)
```



```
000010                          BUF.US  := BUF.UA+1      ;[BDA] BI ADDR BITS 16/30 -> BITS <0,13>
000011                          BUF.GP  := BUF.US+1      ;[BDA] GROUP FOR THIS SECTOR -> BITS <0,7>
                                                         ;[BDA] 2ND SEGMENT WORD COUNT -> BITS <8,15>
000012                          BUF.BC  := BUF.GP+1      ;[BDA] BYTE COUNT FOR U.PROC (TO/FROM HOST)
000013                          BUF.U1  := BUF.BC+1      ;2ND SEGMENT HOST ADDRESS LOW
000014                          BUF.U2  := BUF.U1+1      ;[BDA]2ND SEGMENT HOST ADDRESS HIGH -> BITS <0,13>
                                ;BUF.U3 := BUF.U2+1      ;[BDA] BYTE COUNT FOR SECOND SEGMENT -> BITS <0,8>
000015                          BUF.LL  := BUF.U2+1      ;[BDA]BUFFER CONTROL BLOCK LENGTH

                                ; *-------------------------------------------------------------------------*
                                ; *     DATA BUFFER DEFINITIONS                                             *
                                ; *-------------------------------------------------------------------------*

000000                          BUF.56  := 0             ; START OF 256 WORD DATA BUFFER
000400                          BUF.ED  := BUF.56+SECSZ  ; DATA ERROR CORRECTION CODE (EDC)
000401                          BUF.EC  := BUF.ED+1      ; START OF 12 WORDS HOLDING 17 PACKED ECC SYMBOLS
000415                          BUF.DL  := BUF.EC+12.    ; DATA BUFFER LENGTH

                                ; ********** END BUFFER MAP DEFINITIONS *************

                                ; *-------------------------------------------------------------------------*
                                ; *     SUBUNIT CHARACTERISTICS,                                            *
                                ; *     BUFFER CONTROL BLOCK,                                               *
                                ; *     SDI INTERCONNECT CONTROL BLOCK, AND                                 *
                                ; *     BUFFER ALLOCATION                                                   *
                                ; *-------------------------------------------------------------------------*

000200                          NBCB    := 32.*4         ; [BDA]NUMBER OF BUFFER CONTROL BLOCKS
000051                          NBUFR   := 41.           ; [ch04]Number of data buffers that are dynamically
                                                         ;       allocated.  Buffer NBUFR+1 (ie, buffer 42)
                                                         ;       is reserved for use by the comparison routine.
000010                          NSCB    := 8.            ; NUMBER OF SUBUNIT CHARACTERISTICS BUFFERS
000004                          NSDI    := 4.            ; NUMBER OF SDI INTERCONNECTS
001352                          DM.BEG  := 1352          ; DM MACHINE MUST START AT 1352 !!!!!!!

000000                          ASSUME  PKTEND,LE,DM.BEG; MAKE SURE PACKETS DON'T INTRUDE ON DM SPACE !!!!
001355                          UN.BUF  := DM.BEG+3      ; START OF SUBUNIT CHARACTERISTICS BUFFERS
001355                          SDIBEG  = <UN.BUF&177776>+1; [V05]START OF SDI CONTROL BLOCKS (ODD)

000000                          ASSUME  <SDIBEG&BIT00>,EQ,1; MAKE SURE START IS ODD
001355                          SDI.1   := SDIBEG        ; START OF SDI INTERCONNECT #1 CONTROL BLOCK
001475                          SDI.2   := SDI.1+SDIB.L  ; START OF SDI INTERCONNECT #2 CONTROL BLOCK
001615                          SDI.3   := SDI.2+SDIB.L  ; START OF SDI INTERCONNECT #3 CONTROL BLOCK
001735                          SDI.4   := SDI.3+SDIB.L  ; START OF SDI INTERCONNECT #4 CONTROL BLOCK
002055                          BUFBEG  := SDI.4+SDIB.L  ; START OF BUFFER CONTROL BLOCKS
005255                          BUFEND  := BUFBEG+<NBCB*BUF.LL>; END OF BUFFER CONTROL BLOCKS
005255                          BUFR1.  := BUFEND        ; START OF 1ST DATA BUFFER
005672                          BUFR2.  := BUFR1.+BUF.DL ; START OF 2ND DATA BUFFER
033317                          DATEND  := BUFR1.+<<NBUFR+1>*BUF.DL> ; [ch04]END OF DATA BUFFERS (INCLUDING COMPARE BUFFER)
033317                          STCACH  := DATEND        ; START OF PTE CACHE AREAS
033737                          ENCACH  := STCACH+<NSDI*MEMSZ*2> ; ALLOCATE CACHE FOR EACH SDI

000000                          ASSUME  ENCACH,LT,ALGADR ; MAKE SURE NO OVERWRITTING ECC DATA
034000                          ALGADR  := 34000         ; ANTI-LOG ADDRESS
```

```
036000                          LGADR   := ALGADR+2000  ; LOG ADDRESS
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *      CONTROLLER INTERNAL ERROR CODES                                    *
                                ; *-------------------------------------------------------------------------*

000001                          ER.PRD  := 1                    ; UNIBUS PACKET READ ERROR
000002                          ER.PWR  := 2                    ; UNIBUS PACKET WRITE ERROR
000003                          ER.RP2  := 3                    ; CONTROLLER BI READ PARITY ERROR 2
000003                          ER.WP2  := 3                    ; CONTROLLER BI WRITE PARITY ERROR 2
000003                          ER.BP1  := 3                    ; CONTROLLER BCI PARITY ERROR 1
010000                          ER.SRP  := 10000                ; CONTROLLER BI PARITY ERRORS/SHIFTED FOR LEDS
000004                          ER.RAP  := 4                    ; CONTROLLER RAM PARITY ERROR
020000                          ER.SAP  := 20000                ; CONTROLLER RAM PARITY ERROR/SHIFTED FOR LEDS
000005                          ER.ROP  := 5                    ; CONTROLLER ROM PARITY ERROR
030000                          ER.SOP  := 30000                ; CONTROLLER ROM PARITY ERROR/SHIFTED FOR LEDS
000006                          ER.RRD  := 6                    ; UNIBUS RING READ ERROR
000007                          ER.RWR  := 7                    ; UNIBUS RING WRITE ERROR
000010                          ER.INT  := 8.                   ; UNIBUS INTERRUPT MASTER FAILURE
000011                          ER.HTO  := 9.                   ; HOST ACCESS TIMEOUT ERROR
000012                          ER.NIM  := 10.                  ; HOST EXECEEDED COMMAND LIMIT
000013                          ER.MST  := 11.                  ; UNIBUS BUS MASTER FAILURE
000014                          ER.DMX  := 12.                  ; DM XFC FATAL ERROR
000015                          ER.TMO  := 13.                  ; CONTROLLER HARDWARE ERROR
000016                          ER.VCI  := 14.                  ; INVALID VIRTUAL CIRCUIT IDENTIFIER
000017                          ER.IWR  := 15.                  ; INTERRUPT WRITE ERROR ON UNIBUS
000026                          ER.MRR  := 22.                  ; MAPPING REGISTER READ ERROR
                                                                ; EITHER TIMEOUT OR PARITY WHILE READING
                                                                ; PTES FROM HOST.  See Sec 8.0 UQSSP.
                                ;ER.SUN := 23.                  ; TOO MANY SUB-UNITS ON CONTROLLER kjn
                                                                ; ER.SUN is not referenced
                                                                ; elsewhere in code. 23. reserved for
                                                                ; attempt to map when mapping not supported
                                                                ; sec 8.0 uqssp
000144                          ER.MER  := 100.                 ; BI MASTER ERROR (must be controller specific)
000146                          ER.STP  := 102.                 ; BI STOP OCCURED
000147                          ER.BCA  := 103.                 ; BCAI PARITY TO RAM ERROR
000150                          ER.RTO  := 104.                 ; [mjt08] BI RETRY TIME OUT
000150                          BERMAX  := ER.RTO
000000                                  ASSUME BERMAX,EQ,ER.RTO
                                .PAGE
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *     SYSTEM MACROS                                                       *
                                ; *-------------------------------------------------------------------------*

                                .MACRO  ASSUME  V1,TST,V2               ; ASSUME TWO VALUES MEET CONDITION
                                .if tst,<<V1>-<V2>>
                                .iff

                                    ?ASSUME - VALUES V1 AND V2 ARE NOT TST

                                .endc
                                .ENDM
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *     SDI LEVEL 0 AND 1 EQUATES                                           *
                                ; *-------------------------------------------------------------------------*

152000                          CONTCD  := ^H0D400      ; CONTINUE FRAME CODE
164000                          ECHOCD  := ^H0E800      ; ECHO FRAME CODE
131000                          ENDCD   := ^H0B200      ; END FRAME CODE
046400                          NENDCD  := <<-ENDCD/256.>-1>*256.; NOT END FRAME CODE (SET FOR XOR IN END.F)
025400                          FORICD  := ^H02B00      ; FORMAT ON INDEX CODE
046400                          FORSCD  := ^H04D00      ; FORMAT ON SECTOR CODE
013400                          READCD  := ^H01700      ; SELECT HEAD AND READ CODE
070400                          STRTCD  := ^H07100      ; START FRAME CODE
122400                          WRITCD  := ^H0A500      ; SELECT HEAD AND WRITE CODE
107000                          SGRPCD  := ^H08E00      ; SELECT GROUP CODE
175377                          WRRDX1  := <-<READCD&WRITCD>-1>
131000                          WRRDXR  := <<READCD!WRITCD>&WRRDX1>; READ/WRITE XOR VALUE

                                ; *-------------------------------------------------------------------------*
                                ; *     SDI LEVEL 2 EQUATES                                                 *
                                ; *-------------------------------------------------------------------------*

000176                          COMPLT  := ^H07E        ; COMPLETED OP CODE
000201                          CHGMOD  := ^H081        ; CHANGE MODE OP CODE
000202                          CHGFLG  := ^H082        ; CHANGE CONTROLLER FLAGS OP CODE
000001                          DCN.ST  := BIT00        ; DISCONNECT SPIN DOWN FLAG
000200                          DCN.TT  := BIT07        ; DISCONNECT TERMINATE TOPOLOGY FLAG
000204                          DISCON  := ^H084        ; DISCONNECT OP CODE
000005                          DRVCLR  := ^H005        ; DRIVE CLEAR OP CODE
000006                          ERECOV  := ^H006        ; ERROR RECOVERY OP CODE
000207                          GTCCHR  := ^H087        ; GET COMMON CHARACTERISTICS OP CODE
000170                          GTCRSP  := ^H078        ; GET COMMON CHARACTERISTICS RESPONSE CODE
000011                          GETSTA  := ^H009        ; GET STATUS OP CODE
000210                          GTUCHR  := ^H088        ; GET SUBUNIT CHARACTERISTICS OP CODE
000167                          GTURSP  := ^H077        ; GET SUBUNIT CHARACTERISTICS RESPONSE CODE
000012                          INSEEK  := ^H00A        ; INITIATE SEEK OP CODE
000213                          ONLINE  := ^H08B        ; ONLINE OP CODE
000014                          RUNN    := ^H00C        ; RUN OP CODE
000216                          RECALB  := ^H08E        ; RECALIBRATE OP CODE
000003                          SDIRTY  := 3            ; SDI LEVEL TWO ERROR RETRY COUNT + 1
000200                          SDTIMO  := 200          ; SERDES TIMEOUT
000006                          SEEKL   := 6            ; LENGTH OF SDI LEVEL 2 SEEK COMMAND
000366                          STARSP  := ^H0F6        ; GET STATUS RESPONSE
000220                          TOPOL   := ^H090        ; TOPOLOGY OP CODE
000157                          TOPRSP  := ^H06F        ; TOPOLOGY RESPONSE
000017                          SDITMO  := 15.          ; CONTROLLER TIMEOUT VALUE (15 SECONDS)
000175                          UNSUCC  := ^H07D        ; UNSUCCESSFUL OP CODE

                                ; *-------------------------------------------------------------------------*
                                ; *     SDI ERROR AND OTHER EQUATES                                         *
                                ; *-------------------------------------------------------------------------*

000010                          CSERR   := BIT03        ; CHECKSUM ERROR ON LEVEL 1 READ
000001                          DCERR   := BIT00        ; DRIVE CLOCK TIMEOUT ON LEVEL 1 READ/WRITE
```

```
000004                          FRERR   := BIT02        ; FRAMING ERROR ON LEVEL 1 READ
000020                          LNERR   := BIT04        ; RESPONSE LENGTH ERROR
000002                          SFERR   := BIT01        ; FIRST WORD NOT START FRAME ERROR
000040                          UNERR   := BIT05        ; UNSUCCESSFUL RESPONSE ERROR
                                .PAGE
```



```
                                .SBTTL  KDB D-PROC DIAGNOSTICS

                                ;;*****************************************************************************
                                ;
                                ;       DIAGNOSTIC HISTORY
                                ;
                                ;;*****************************************************************************

                                ;16-JAN-81       MATT    NPR FIX
                                ;20-JAN-81       CURT    FIX ZERO BUFFER,SAVE FAILURE
                                ;27-JAN-81       MATT    FIXED TWO STEP BITS SET IN SA REG
                                ;29-JAN-81       MATT    SDI TEST FIX
                                ;30-JAN-81       MATT    TEST CRTDS BIT IN DCR
                                ;12-FEB-81       MATT    LED CORRECTION
                                ;18-FEB-81       BILL    BURST VALUE CORRECTION
                                ;25-FEB-81       MATT    NPR ERROR REPORTING CORRECTION
                                ;27-MAR-81       MATT    IMPLEMENT RICHY'S CODE SAVING
                                ;17-JUN-81       MATT    CODE NOT TO TOUCH FAILUR WORD
                                ;18-JUN-81       MATT    CORRECTIONS OF PREVIOUS EDIT
                                ;19-JUN-81       MATT    PURGE/POLL WIPES R0 FIXED
                                ;24-JUN-81       CURT    ADD SIGNITURE ANALYSIS CODE - 1 WORD
                                ;24-JUN-81       MATT    TOOK EXTRA WORD OUT OF ERROR ROUTINE/COMMENT CHANGE
                                ;30-JUN-81       BILL    REMOVE LAST FAILURE STORE
                                ;16-JUL-81       BILL    FIX CROM TIMEOUT VALUES
                                ;03-AUG-81       MATT    FIX STEP 1 DISPLAY ERROR
                                ;14-AUG-81       MATT    ACLO FIX(ONE LAST TIME)
                                ;17-NOV-81       MATT    FROM 4K TO 16K
                                ;16-FEB-82       MATT    SDI CHANGES FOR 8 PORT DIAGNOSTIC MODE
                                ;23-FEB-82       MATT    ADD '+DRINIT' AT STEP4:
                                ;05-MAY-82       MATT    TOOK OUT DIAPRT -- FOR ALL REFERENCES SEARCH FOR ;MJT(SDI)
                                ;14-MAY-82       MATT    FIXED MAX RING LEN NOT CLEARING RING AREA PROBLEM
                                ;28-MAY-82       MATT    FIXED WRAP FEATURE, LEDS FLASH ;MJT1
                                ;18-JUN-82       CURT    UDA52 NEW BOARD CHANGES
                                ;19-JUN-82       CURT    UDA52 NEW BOARD CHANGES - NOW IT WORKS
                                ;23-JUL-82       MATT    CHANGE DCRTST TO REPORT BOARD 2 ERROR
                                ;23-NOV-82       MATT    COMMENTED OUT U&D CR TESTS AS CODE REDUCTION ; [U52EC1]
                                ;04-FEB-83       MATT    SPLIT UP U AND D PROC DIAGNOSTICS
                                ;10-OCT-83       MATT    START CHANGING CODE FOR REAL QDA
                                ;18-OCT-83       MATT    CONTINUE TO CHANGE FOR REAL QDA
                                ;01-MAR-1984     MIKE    CONTINUE TO CHANGE FOR REAL UDA/QDA
                                ;24-APR-1984     MIKE    SDI PORT WRAP TEST & DFAIL U-CROM TEST (BL0)
                                ;04-MAY-1984     MIKE    DEFINE QB,MP FOR STEP 1/MOVE DER TO UER IN ERROR ROUTINE.
                                ;14-MAY-1984     MIKE    PUT '.ORG 700' AT END OF DIAGNOSTIC SECTION.
                                ;30-MAY-1984     MIKE    CONDITIONED OUT 1 SECOND TIMEOUT CODE IN DDIAG. (BL2)
                                ;24-AUG-1984     MIKE    ADDED LOOP-ON-TEST CODE, VERIFIED & CORRECTED ERROR CODES
                                ;27-AUG-1984     MIKE    ADDED & CORRECTED ERROR CODES
                                ;11-SEP-1984     MIKE    ADDED SA ERROR CODES TO ROUTINES & ERROR CALLS
                                ;21-SEP-1984     MIKE    GENERAL ROUTINE CLEAN-UP & ADDING C$TYPE CONDITIONAL
                                ;                         FOR BDA DEVELOPMENT
                                ;08-OCT-1984     MIKE    GENERAL ROUTINE CLEAN-UP & ADDED PRELIMINARY BDA CODE.
                                ;22-OCT-1984     MIKE    DELETED A FILLER INSTRUCTION FROM 'STEP' ROUTINE
                                ;05-NOV-1984     MIKE    ADD COMMENT TO ERROR ROUTINE DESCRIPTION,
                                ;                         CHANGE LITERAL IN RAM BUFFER TEST &
                                ;                         CLEANUP OF COMMENTS IN SDI TEST
```

```
                                ;08-NOV-1984    MIKE    SOME COMMENT & RAM BUFFER TEST CHANGES
                                ;12-NOV-1984    MIKE    SOME COMMENT CHANGES
                                ;26-JUL-1985    MIKE    KBD uCODE FREEZE HERE...
                                ;11-DEC-1985    MIKE    REMOVE ANY REFERENCES TO BI STOP TEST                   [mr1001]
                                ;08-JAN-1987    MIKE    REMOVE REFERENCE TO BI FAST SELF-TEST IN KDA BUILD      [mr1002]
```



```
                                ;
                                ;       DIAGNOSTIC SPECIALLY USED REGISTERS
                                ;

000000                          LT400   = 0             ;0 = OLD CODE
000001                          UDA1    = 1             ;1 = NEW UDA1 ETCH

000040                          SM      = BIT05         ;STEP1 SPECIAL MODE
000100                          MP      = BIT06         ;STEP1 MAPPING SUPPORTED
000400                          DIAG    = BIT08         ;STEP1 ENHANCED DIAGNOSTICS SUPPORTED
000400                          DI      = BIT08         ;STEP1 ENHANCED DIAGNOSTICS SUPPORTED
001000                          QB      = BIT09         ;STEP1 22 BIT ADDRESSING SUPPORTED

000400                          SF      = BIT08         ;STEP4 SPECIAL FUNCTION BIT

001000                          BOARD2  = BIT09         ;0 = BOARD #1, 1 = BOARD #2
100000                          ERRBIT  = BIT15         ;0 = NO ERROR, 1 = ERROR

                                ;
                                ;       DIAGNOSTIC EQUATES
                                ;

000305                          ECSUML  := 305          ;ECC SUM LOWER BYTE
022000                          ECSUMH  := 22000        ;ECC SUM UPPER BYTE
176000                          ECCMSK  := 176000       ;ECC MASK

000003                          SCLR    :=  3           ; SDI - SET SDI INFC I/O CLEAR
000013                          RCLR    := 13           ; SDI - RESET SDI INFC I/O CLEAR
```

```
;;*****************************************************************************
;
;       Diagnostic test sequence, with respect to time and each processor
;
;       U-PROC (TEST LABEL)                 D-PROC (TEST LABEL)
;       -------                             -------
;
;       U.RSTRT:                            D.RSTRT:
;
;       SEQUENCER RELIABILITY               SEQUENCER RELIABILITY
;       SEQUENCER STACK TEST                HANG ON 1 INSTRUCTION (TSTHNG)
;       ALU TEST (ALUTST)                      HANG D-PROC (TSTHNG)
;       CONTROL REG TEST (UCRTST)                 "       (TSTHNG)
;       DFAIL TEST                          RUNS THROUGH ALL OF U and D ROM
;
;       If an error occurs here, the D-PROC will continue to run
;       thru the U or D ROM, while the U-PROC reports the error.
;
;       RELEASE D-PROC (RELES)              SEQUENCER STACK TEST
;          HANG U-PROC (RELES)              RELEASE U-PROC (RELES)
;       DISPLAY LEDS (DSLEDS)                  HANG D-PROC (RELES)
;       ROM PE TEST (UROMPE)                        "       (RELES)
;       RELEASE D-PROC (RELES)                      "       (RELES)
;          HANG U-PROC (RELES)              ALU TEST (ALUTST)
;                "       (RELES)            DISPLAY LEDS (DSLEDS)
;                "       (RELES)            ROM PE and TIMEOUT (DROMPE)
;                "       (RELES)            BOARD 2 TEST (B2TST)
;                "       (RELES)            CONTROL REG TEST (DCRTST)
;                "       (RELES)            RAM PE TEST (DRAMPE)
;                "       (RELES)            RELEASE U-PROC (RELES)
;       RAM PE TEST (URAMPE)                   HANG D-PROC (RELES)
;       BI FAST SELF-TEST ENABLED ?                 "       (RELES)
;       IF SO,                                      "       (RELES)
;          RELEASE D-PROC,                  BI FAST SELF-TEST ENABLED?
;          GOTO FAST TEST (FSTST)           IF SO,
;                                              GOTO FAST TEST (FSTST)
;       RELEASE D-PROC (RELES)              HANG D-PROC (RELES)
;          HANG U-PROC (RELES)              RAM BUFFER TEST (RAMTST)
;                "       (RELES)            RELEASE U-PROC (RELES)
;       RAM BUFFER TEST (RAMTST)               HANG D-PROC (RELES)
;       RELEASE D-PROC (RELES)                      "       (RELES)
;          HANG U-PROC (RELES)              SDI TEST (SDITST)
;                "       (RELES)            SERDES WRITE (SDWRT)
;                "       (RELES)            SERDES READ (SDRD)
;                "       (RELES)            ECC TEST (ECCRD)
;                "       (RELES)            RELEASE U-PROC (RELES)
;
;                              Continue...
```



```
;
;       U-PROC (TEST LABEL)                 D-PROC (TEST LABEL)
;       -------                             -------
;
;       FSTST:                              FSTST:
;
;       SET BI LOOPBACK MODE                   HANG D-PROC (RELES)
;       BCAI BUFFER TEST (BCATST)                   "       (RELES)
;       BI PE TEST (BIPE) *PART MFG*                "       (RELES)
;       BIIC BUFFER TEST (BIITST)                   "       (RELES)
;       GET BI NODE ID (GETID)                      "       (RELES)
;       SET BI NORMAL MODE                          "       (RELES)
;       BIIC BUFFER TEST (BIITST)                   "       (RELES)
;       POLL TEST (POLTST)                          "       (RELES)
;       BI MEMORY TEST (BIMEM) *MFG*                "       (RELES)
;       LOAD BIDTYP REGISTER                        "       (RELES)
;       CLEAR BROKE BIT IN BICSR                    "       (RELES)
;       SET STEP 1 IN SA REGISTER                   "       (RELES)
;
;       U.END:                              D.END:
;
;       WAIT 50ms FOR HOST RESPONSE                 "       (RELES)
;       REDO DIAGNOSTIC ???
;       IF SO,                              IF SO,
;          SET FLAG, RELEASE D-PROC,           GOTO RESTART (D.RSTRT)
;          GOTO RESTART (U.RSTRT)
;       GOTO INITIALIZATION                 GOTO D-PROC IDLE LOOP
;       STEP.1,                             INITIALIZE DRIVES
;       STEP.2,
;       STEP.3,
;       DMA TEST (DMATST),
;       STEP.4,
;       GOTO U-PROC IDLE LOOP.
;
;;*****************************************************************************
```

```
                                ;;*****************************************************************************
                                ;
                                ; LEV05 ** USES THE ENTIRE STACK **
                                ;
                                ;       The 2911 sequencers are tested with CONTINUES, JUMPS, CALLS and
                                ;       RETURNS to make sure they are functioning properly.  The STACK is
                                ;       tested to make sure that it's 4 locations can be wrapped around.
                                ;
                                ;       When the diagnostic is restarted, the D-PROC will be within 2 cycles
                                ;       of the U-PROC.  If it isn't, a race with the registers may occur and
                                ;       cause an error.
                                ;
                                ;       REGISTERS USED:
                                ;               Q  (TSTCNT) is used as the test counter.
                                ;               R3 (HANG) holds the SA register STEP bits until the D-PROC
                                ;                hangs.  Then, this register becomes a hand shaking mechanism
                                ;                between the U-PROC and the D-PROC, to coordinate the HANG/
                                ;                RELEASE process.
                                ;               R11 (DER) is used as the D-PROC error register.
                                ;               R17 is used as the STACK counter.
                                ;
                                ;;*****************************************************************************

000000  120457  007477  010000          INC     R17\0,,BAR      @SCLR           ;WHEN RESET, INCREMENT R17 AND DISPLAY ON
                                                                                ; DATA BUS. FALL THROUGH TO RESET TIMERS
000001  013440  000000  000000          NOP                                     ;NEEDED TO KEEP U-PROC AND D-PROC ALIGNED ON
                                                                                ; POWER-UP OR WHEN SST IS ENABLED
000002  034457  004017  130005  D.RSTRT: CLR     R17,,DCRD       %CALL JMPTST    ;CLEAR DCRD, R17.
                                                                                ; PUSH SEQERR ADDRESS ONTO STACK FOR
                                                                                ; POTENTIAL PARITY ERRORS.

                                ;;*****************************************************************************
                                ;
                                ;       THE SEQUENCE ERROR CODE IS EXECUTED IN AN ATTEMPT TO CONTROL A FATAL
                                ;       SEQUENCER ERROR.
                                ;
                                ; NOTE: Fatal sequencer errors, try to display 104000 somewhere.
                                ;
                                ;;*****************************************************************************

000003  133750  000210  120004  SEQERR: MOV     #ERR00,UDDI     %CALL    .+1    ;SEQUENCER ERROR
000004  034444  004004  100003          CLR     CRI,,DCRD       %JMP SEQERR     ;DISPLAY ALL LEDS /
                                                                                ; JUMP TO SEQUENCER ERROR.

000005                                  ASSUME HANG,EQ,R3

000005  034443  000603  000003  JMPTST: CLR     HANG    @RDPF   %JUMPN  SEQERR  ;ERROR IF THIS JUMP WORKS /
                                                                                ; RESET D-PROC FLAG TO INDICATE
                                                                                ; NO LOOP ON TEST.
000006  013440  000000  030003          NOP                     %CUMPN  SEQERR  ;ERROR IF THIS CALL WORKS.
000007  004240  000660  100015          CLR     TSTCNT  @RCLR   %JMP    JUMP    ;CLEAR TSTCNT(Q) / RESET I/O CLEAR /
                                                                                ; TRY FIRST JUMP.
000010  133750  000210  130003          MOV     #ERR00,UDDI     %CALL   SEQERR  ;ERROR IF JUMP DID NOT WORK.

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```



```
                                ;;*****************************************************************************
                                ;
                                ;       RETURN ROUTINE
                                ;
                                ;;*****************************************************************************

000011  013440  000000  127777  STEP:   NOP                     %RET            ;TRY FIRST RETURN.
000012  133750  000210  100003          MOV     #ERR00,UDDI     %JMP    SEQERR  ;ERROR IF RETURN DID NOT WORK.

                                ;;*****************************************************************************
                                ;
                                ;       CROM PARITY ERROR
                                ;
                                ;;*****************************************************************************

000013  013440  000000  137777  TSCRPE: .WORD   13440,0,137777                  ;FORCE CROM PARITY ERROR WHEN
                                                                                ; DFAIL BIT IS SET BY U-PROC.

                                ;;*****************************************************************************
                                ;
                                ;       D-PROC WAIT HERE WHILE U-PROC RUNNING SEQUENCE TEST. DFAIL TEST WILL
                                ;       CAUSE D-PROC TO GO ALL THROUGH MEMORY THEN START EXECUTION AT
                                ;       'SEQTST:'. THE D-PROC CAN STAY HERE ABOUT 116.66 MSEC (380 MILLION
                                ;       INSTRUCTIONS) BEFORE TIMEOUT WILL OCCUR.
                                ;
                                ;;*****************************************************************************

000014  113740  004380  110014  TSTHNG: MOV     #LEDS,\N,DCRD   %JMP    TSTHNG  ;LET D HANG HERE UNTIL
                                                                                ; U-PROC SETS DFAIL BIT.

                                ;;*****************************************************************************
                                ;
                                ;       TEST RETURN
                                ;
                                ;;*****************************************************************************

000015  013440  000000  120011  JUMP:   NOP                     %CALL   STEP    ;CALL ROUTINE TO SEE IF RETURN WORKS
                                                                                ; RETURN WORKED.
000016  013440  000000  110014          NOP                     %JMP    TSTHNG  ;JUMP TO HANG FOR DFAIL IN D-CROM.
000017  013440  000000  110014          NOP                     %JMP    TSTHNG  ;JUMP TO HANG FOR DFAIL IN U-CROM.
000020  013440  000000  000000          NOP                                     ;NOP TO SYNC U & D PROCESSORS
000021  013440  000000  000000          NOP                                     ; IN THE DFAIL TEST

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```

```
                                ;;*****************************************************************************
                                ;
                                ;       TEST STACK WRAP AROUND
                                ;
                                ;;*****************************************************************************

000022                                  ASSUME TSTHNG+6,EQ,SEQTST
                                                                                ;IF WE GET HERE, BOTH JMP, CALL & RET
                                                                                ; WORKED, ELSE
                                                                                ; ALU ERROR WILL RETURN TO 'SEQERR'.
000022  033757  000004  100025  SEQTST: MOV     #4,R17          %JMP    2$      ;SET STACK PUSH COUNT
                                                                                ; LOOP HERE FOREVER IF ALU
                                                                                ; STACK OR NZRO FAILED.
000023  031477  010017  070023  1$:     DEC\T   R17             %CNZRO  1$      ;PUSH '.+1' ON STACK 3 TIMES.
000024  030557  000010  137777          ADD     #8.,R17         %RET            ;RETURN HERE 3 TIMES, THEN
                                                                                ; RETRUN TO '2$+1'.
000025  031457  000017  130023  2$:     DEC     R17             %CALL   1$      ;PUSH REAL RETURN ADDRESS
                                                                                ; STACK TEST RETURN HERE
000026  016557  010070  100003          XOR     #56.,R17\N      %JZRO   SEQERR  ;ERROR IF R17 IS ZERO /
                                                                                ; GO THRU HERE TWICE,
                                                                                ; 1st R17= 32. & 2nd R17= 56.
000027  033753  010004  037777          MOV     #4,R13          %RNZRO          ;TEST STACK WRAP-AROUND
                                                                                ; EQUALS RETURN TO '1$+1'
                                                                                ; STAY THERE 3 TIMES UNTIL STACK
                                                                                ; WRAPS AROUND.
                                                                                ; IF WE GET HERE, WRAP-AROUND WORKS
                                                                                ; WORKS, ELSE RETURN TO 'SEQERR'.
000030  031453  000013  130052  SEQ01:  DEC     R13             %CALL   PSTACK  ;PUSH STACK FULL OF ERROR
                                                                                ; RETURN ADDRESSES.
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEV00 WILL HANG HERE
                                ;
                                ;       ERROR ROUTINES
                                ;
                                ;       Red LED codes displayed;
                                ;               9 (hex) = BOARD 1 BAD
                                ;               A (hex) = BOARD 2 BAD
                                ;
                                ;       Yellow LED codes displayed;
                                ;               ON  = NODE PASSED SELF-TEST
                                ;               OFF = NODE FAILED SELF-TEST
                                ;
                                ;       Format of error code that will be put into the SA register;
                                ;
                                ;               15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
                                ;               --+--------+--------+--------+--------+--------
                                ;               E  S  S  S  S  P   B  T           T
                                ;               R  T  T  T  T  R   O     S           S
                                ;               R  E  E  E  E  O   A        T           T
                                ;               B  P  P  P  P  C   R           CODE           COUNT
                                ;               I  4  3  2  1      D
                                ;               T
                                ;                                  ^    ^
                                ;                                  !    !
                                ;                                  !    +------ 0= Board 1 (PROC) at fault
                                ;                                  !    +------ 1= Board 2 (SDI)  at fault
                                ;                                  !
                                ;                                  +---------- 0= D-PROC detected error
                                ;                                  +---------- 1= U-PROC detected error
                                ;
                                ;;*****************************************************************************

                                        ;;BOARD 1 ERROR - RAMPE/CROMPE/TIMEOUT RETURN ADDRESS

000031  004240  000000  000000  ERRB1A: CLR     TSTCNT                          ;ALU TEST ERROR/TSTCNT MAY HAVE GARBAGE
000032  033751  000040  000000  ERRB1E: MOV     #ERR01,DER                      ;CONTROL PE/ALU TEST
000033  133551  000010  000000          BIS     #STEP1,DER                      ;SET STEP 1 TO REPORT BRD1 ERROR
000034  133744  000144  110042          MOV     #<LED4+LED2+DFAIL>,CRI %JMP ERRSET ;LEDS '0110' 9 (HEX) 0=ON
```

```
                                        ;;BOARD 2 ERRORS

000035  033751  000100  100040  ERRB2E: MOV     #ERR02,DER      %JMP    ERRB2   ;RAM PE/RAM BUFFER/RAM ERROR (ANY PROCESSOR)
000036  033751  000140  110040  ERRB2H: MOV     #ERR03,DER      %JMP    ERRB2   ;SDI/SERDES/ECC ERROR
000037  033751  000200  100040  ERRB2I: MOV     #ERR04,DER      %JMP    ERRB2   ;CONTROL REG TEST ERROR

000040  133551  000012  010000  ERRB2:  BIS     #<STEP1+BOARD2>,DER             ;SET STEP 1 TO REPORT BRD2 ERROR
000041  133744  000124  000000          MOV     #<LED4+LED1+DFAIL>,CRI          ;LEDS '0101' A (HEX) 0=ON

                                        ;;ERRSET is the diagnostic error trigger point...

000042  133551  000200  130043  ERRSET: BIS     #ERRBIT,DER     %CALL   1$      ;SET ERROR BIT IN ERROR REG
                                                                                ; SETUP RETURN ADDR IF VECTOR.
000043  033051  000011  000000  1$:     BIS     TSTCNT,DER                      ;SET TSTCNT(Q) IN ERROR REG
000044  033447  000011  130156          MOV     DER,UER         %CALL   WAIT    ;SEND ERROR STATUS TO U-PROC ERROR REG /
                                                                                ; WAIT, BI SELF-TEST MAY NOT BE COMPLETE.
000045  034443  000003  000000          CLR     HANG                            ;CLEAR HANG REG SO U-PROC CAN CONTINUE.

000046  033444  004004  000000  DHANG:  MOV     CRI,,DCRD                       ;SET LEDS ON BOARD 2
000047  013451  000011  000000          TST     DER                             ;TEST D-PROC ERROR REG
000050  013440  000000  100046          NOP                     %JMP    DHANG   ;LOOP HERE FOREVER....
```



```
                                ;;*****************************************************************************
                                ;
                                ;       CLEAR TSTCNT(Q) BECAUSE IT COULD BE FILLED WITH GARBAGE AT TIME
                                ;       OF INIT.
                                ;
                                ;;*****************************************************************************

000051  004240  000000  127777  CLRTST: CLR     TSTCNT          %RET            ;TSTCNT(Q) = 0 (COULD BE GARBAGE)

                                ;;*****************************************************************************
                                ;
                                ;       PUSH STACK FOR STACK WRAP TEST
                                ;
                                ;;*****************************************************************************

000052  000057  010017  040030  PSTACK: ADD\F   R17,TSTCNT      %JNZRO  SEQ01   ;JUMP BACK UNTIL STACK FULL, ELSE
                                                                                ; INCREMENT TSTCNT(Q).
```

```
                                ;;*****************************************************************************
                                ;
                                ;       The D-PROC calls its BOARD 1 diagnostics
                                ;
                                ;;*****************************************************************************

                                D.DIAG:
                                HD.T0:
                                HD.T1:
000053  133544  004003  120136  HD.T2:  BIS     #<0DDP+INDIAG>,CRI,DCRD %CALL RELES ;SET ODD PE & INDIAG BITS FOR U-PROC,
                                                                                    ;AND HANG D-PROC, LET U-PROC DO UROMPE TEST.

000054  037160  010000  160166  T3:     XNOR\T  R0,R0           %CZRO   ALUTST  ;INITIALIZE R0 TO -1 / DO ALU TEST
000055  013460  010000  050131          NOP\T                   %JNZRO  D.END   ;JUMP IF ERROR
000056  014471  060011  150054          CLR\T   DER,\N          %JDPF   T3      ;LOOP IF D-PROC FLAG SET

000057  013460  010000  170151          NOP\T                   %CZRO   DSLEDS  ;ROTATE LED DISPLAY

000060  133764  014363  160266  T4:     MOV\T   #<LEDS+0DDP+INDIAG>,CRI,DCRD %CZRO DROMPE ;RESTORE CRI REG /
                                                                                    ; DO ROM PARITY ERROR TEST
000061  013460  010000  050131          NOP\T                   %JNZRO  D.END   ;JUMP IF ERROR
000062  014471  060011  140060          CLR\T   DER,\N          %JDPF   T4      ;LOOP IF D-PROC FLAG SET
```



```
                                ;;*****************************************************************************
                                ;
                                ;       The D-PROC calls its BOARD 2 diagnostics
                                ;
                                ;;*****************************************************************************

000063  013460  010000  170161          NOP\T                   %CZRO   B2TST   ;SEE IF BOARD 2 IS THERE

000064  013460  010000  170251  T5:     NOP\T                   %CZRO   DCRTST  ;DO CONTROL REGISTER TEST
000065  013460  010000  050131          NOP\T                   %JNZRO  D.END   ;JUMP IF ERROR
000066  014471  060011  150064          CLR\T   DER,\N          %JDPF   T5      ;LOOP IF D-PROC FLAG SET

000067  133764  014363  160304  T6:     MOV\T   #<LEDS+0DDP+INDIAG>,CRI,DCRD %CZRO DRAMPE ;RESTORE DCR /
                                                                                    ; DO RAM PARITY ERROR TEST
000070  013460  010000  160136          NOP\T                   %CZRO   RELES   ;HANG D-PROC, LET U-PROC DO RAM PE TEST
000071  013460  010000  050131          NOP\T                   %JNZRO  D.END   ;JUMP IF ERROR
000072  014471  060011  150087          CLR\T   DER,\N          %JDPF   T6      ;LOOP IF D-PROC FLAG SET
000073  013444  000004  000000          TST     CRI                             ;ARE WE IN BI FAST SELF-TEST MODE?
000074  013447  030007  010114          TST     UER             %JNNEG  FSTST   ;JUMP IF YES /
                                                                                ; DID THE U-PROC HAVE AN ERROR?

000075  013460  010000  170322  T7:     NOP\T                   %CZRO   RAMTST  ;RAM BUFFER TEST
000076  013460  010000  050131          NOP\T                   %JNZRO  D.END   ;JUMP IF ERROR
000077  014471  060011  150075          CLR\T   DER,\N          %JDPF   T7      ;LOOP IF D-PROC FLAG SET

000100  013460  010000  160136  HD.T8:  NOP\T                   %CZRO   RELES   ;HANG D-PROC, LET U-PROC DO RAM BUFFER TEST

000101  035564  014317  160354  T9:     BIC\T   #<RAMPE+0PM+17>,CRI,DCRD %CZRO SDITST ;SDI INTERCONNECT TEST / DISABLE RAM PE &
                                                                                    ; ENABLE DIAGNOSTIC WRAP PORT.
000102  033544  010010  040131          BIS\F   #10,CRI         %JNZRO  D.END   ;JUMP IF ERROR, ELSE
                                                                                ; START TESTING WITH SDI PORT 3.
000103  013440  050000  100111          NOP                     %JNTEST 2$      ;JUMP IF NOT LOOPING ON SDI PORTS
000104  013440  050000  100111          NOP                     %JNTEST 2$      ;JUMP IF NOT LOOPING ON SDI PORTS
000105  003740  000012  010000  1$:     MOV     #12,TSTCNT                      ;LOAD TSTCNT(Q).
000106  033444  004004  130354          MOV     CRI,,DCRD       %CALL SDITST    ;RESTORE CRD / DO SDI PORT TEST
000107  054544  000017  000000          AND\R   #17,CRI                         ;SAVE PORT SELECT BITS /
                                                                                ; SELECT NEXT PORT FOR TEST.
000110  133544  010363  000105          BIS     #<LEDS+0DDP+INDIAG>,CRI %JNZRO 1$ ;JUMP IF NOT DONE TESTING PORTS /
                                                                                  ; RESTORE CRI
000111  013451  000011  000000  2$:     TST     DER                             ;ANY ERRORS?
000112  014471  060011  140101          CLR\T   DER,\N          %JDPF   T9      ;LOOP IF D-PROC FLAG SET.
000113  034464  010004  150114          CLR\T   RLL             %TZRO           ;END DIAGNOSTIC MODE IF NO ERROR
```

```
                                ;;*****************************************************************************
                                ;
                                ;       The D-PROC waits for BI testing to complete on BOARD 1
                                ;
                                ;;*****************************************************************************

000114  133744  010363  000131  FSTST:  MOV     #<LEDS+ODDP+INDIAG>,CRI %JNZRO D.END ;JUMP IF ERROR /
                                                                                ; RESTORE CONTROL REG IMAGE
000115  033444  004004  130136          MOV     CRI,,DCRD       %CALL   RELES   ;HANG D-PROC,
                                                                                ; LET U-PROC INIT RAM W/ ODD PARITY &
                                                                                ; WAIT FOR U-PROC TO CAUSE WR PE2 VECTOR (7762)
000116  013440  000000  120524          NOP                     %CALL   TSTRTN  ;WASTE SOME TIME TO SYNC UP WITH THE U-PROC
000117  013440  000000  120524          NOP                     %CALL   TSTRTN  ; AFTER DOING VECTOR DURING BCI PARITY TEST.
000120  013440  000000  120524          NOP                     %CALL   TSTRTN  ;
000121  013440  000000  120524          NOP                     %CALL   TSTRTN  ;
000122  013447  000007  000000          TST     UER                             ;DID THE U-PROC HAVE AN ERROR?
000123  013440  010000  120136          NOP                     %CZRO   RELES   ;HANG D-PROC AGAIN,
                                                                                ; WAIT FOR U-PROC TO CAUSE RD PE2 VECTOR (7761)
000124  013440  000000  120524          NOP                     %CALL   TSTRTN  ;WASTE SOME TIME TO SYNC UP WITH THE U-PROC
000125  013440  000000  120524          NOP                     %CALL   TSTRTN  ; AFTER DOING VECTOR DURING BCI PARITY TEST.
000126  013447  000007  000000          TST     UER                             ;DID THE U-PROC HAVE AN ERROR?
000127  013440  010000  120142          NOP                     %CZRO   TSTXL   ;GO TEST EXT.TEST_L CONDITION,
                                                                                ; HANG D-PROC AGAIN,
                                                                                ; LET U-PROC DO REST OF BI TESING
000130  034464  010004  140131          CLR\T   RLL             %TZRO           ;END DIAGNOSTIC MODE IF NO ERROR

                                ;;*****************************************************************************
                                ;
                                ;       D END...
                                ;
                                ;;*****************************************************************************

000131  013447  000007  000000  D.END:  TST     UER                             ;DID THE U-PROC HAVE AN ERROR?
000132  013451  010011  050046          TST\F   DER             %JNZRO  DHANG   ;IF SO, HANG THE D-PROC, ELSE
                                                                                ; DID THE D-PROC HAVE AN ERROR?
000133  013240  010000  050040          TST\F   TSTCNT          %JNZRO  ERRB2   ;IF SO, REPORT SOMETHING WENT WRONG WITH BD 2
                                                                                ; WAS TSTCNT(Q) CLEARED BY U-PROC?
000134  133751  010361  100002          MOV     #<LEDS+ODDP>,R11 %JZRO  D.RSTRT ;REDO DIAGNOSTICS IF TSTCNT=0 /
                                                                                ; LEDS OFF WITH ODD PARITY ON.
000135  033551  004300  100701          BIS     #<RAMPE+OPM>,R11,DCRD %JMP D.IDLE ;ENABLE RAM PE /
                                                                                ; GOTO REGULAR FIRMWARE.
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEV01
                                ;
                                ;       RELEASE THE U-PROC FROM HANGING AND LET D-PROC HANG.  RETURN WILL OCCUR
                                ;       WHEN THE HANG REGISTER (R3) IS CLEARED.
                                ;
                                ;       Note:   The U-PROC ERROR REGISTER (R7) will be tested on return.
                                ;
                                ;;*****************************************************************************

000136  034443  000003  120524  RELES:  CLR     HANG            %CALL   TSTRTN  ;RELEASE U-PROC AND
                                                                                ; WASTE ONE INSTRUCTION.
000137  033743  000001  000000          MOV     #1,HANG                         ;KEEP SOMETHING IN HANG
000140  013447  010007  137777  1$:     TST     UER             %RZRO           ;IF ZERO, TEST U-PROC ERROR / RETURN
000141  013443  000003  110140          TST     HANG            %JMP    1$      ;SEE IF PROC SHOULD LEAVE LOOP?

                                ;;*****************************************************************************
                                ;
                                ; LEV01
                                ;
                                ;       This routine tests the EXT.TEST_L condition to determine if the
                                ;       diagnostic should use a special RELEASE routine for executing the BI
                                ;       or Q Bus MEMORY test in the manufacturing only code.
                                ;
                                ;       IF EXT.TEST_L CONDITION IS ALWAYS ENABLED OR DISABLED, DO THE NORMAL
                                ;       RELEASE OF THE U-PROC AND LET D-PROC HANG BY JUMPING TO THE 'RELES'
                                ;       ROUTINE.  RETURN TO THE MAIN PROGRAM FLOW WHEN THE HANG REGISTER (R3)
                                ;       IS CLEARED.
                                ;       IF EXT.TEST_L CONDITION IS TOGGLING (indicating mfg mode), DO NORMAL
                                ;       RELEASE OF THE U-PROC AND LET D-PROC HANG BY CALLING THE 'RELES'
                                ;       ROUTINE.  RETURN TO THIS ROUTINE WHEN THE HANG REGISTER (R3) IS
                                ;       CLEARED.  NOW WAIT FOR THE U-PROC TO COMPLETE THE BI MEMORY TEST BY
                                ;       WAITING FOR R11 (DER) TO BE NON-ZERO.
                                ;
                                ;;*****************************************************************************

000142  013440  050000  100144  TSTXL:  NOP                     %JNTEST 1$      ;CHECK EXT.TEST_L FOR TOGGLE MODE
000143  013440  050000  010136          NOP                     %JTEST  RELES   ;RELEASE IF EXT.TEST_L DID NOT TOGGLE AND
                                                                                ; RETURN FROM RELEASE ROUTINE.
000144  013440  050000  100136  1$:     NOP                     %JNTEST RELES   ;RELEASE IF EXT.TEST_L DID NOT TOGGLE AND
                                                                                ; RETURN FROM RELEASE ROUTINE.
000145  013440  000000  130136          NOP                     %CALL   RELES   ;HANG D-PROC AGAIN,
                                                                                ; LET U-PROC DO SOME BI TESING AND
                                                                                ; RETURN HERE TO DO GET INTO THE
000146  013451  000011  000000          TST     R11                             ; SPECIAL WAIT LOOP.
000147  034471  010011  040136  2$:     CLR\T   R11             %JNZRO  RELES   ;IF ZERO, TEST U-PROC ERROR / RETURN
000150  013451  000011  100147          TST     R11             %JMP    2$      ;WE ARE IN A SPECIAL TEST MODE, SO
                                                                                ; WAIT FOR R11 TO CHANGE

                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;       ROTATE LED DISPLAY
```

```
                                ;               TO SHOW THAT THEY DO FLASH AND THAT THE CODE IS WORKING
                                ;
                                ;       LEDS     8       4       2       1
                                ;                off     off     off     on
                                ;                off     off     on      off
                                ;                off     on      off     off
                                ;                on      off     off     off
                                ;
                                ;       CALLS WAIT
                                ;
                                ;       Input:
                                ;               CR has leds set off & in diagnostic mode.
                                ;
                                ;;*****************************************************************************

000151  133742  000020  000000  DSLEDS: MOV     #<LED1>,R2                      ;START WITH LED#1
000152  013471  030011  167777  1$:     TST\T   DER             %RNEG           ;JUMP IF DONE, ELSE
000153  035144  004002  130156          BIC     R2,CRI,DCRD     %CALL   WAIT    ;LED ON
000154  033144  004002  000000          BIS     R2,CRI,DCRD                     ;LED OFF
000155  073442  000002  110152          SHF\L   R2              %JMP    1$      ;NEXT LED
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEV02
                                ;       WAIT
                                ;
                                ;       WAIT APPROXIMATELY 10ms, BASED ON 346ns SEQUENCER CLOCK
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               R0 is changed.
                                ;
                                ;;*****************************************************************************

000156  133740  000161  000000  WAIT:   MOV     #70400,R0                       ;INITIALIZE LOOP TO 10.ms COUNT
000157  031440  010000  010157  1$:     DEC     R0              %JNZRO  1$      ;KEEP LOOPING TIL DONE
000160  013440  000000  127777          NOP                     %RET            ;RETURN

                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;       B2TST
                                ;               Test for the existence of SDI board (BOARD 2).
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               none
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105102 - No board 2 error
                                ;
                                ;;*****************************************************************************

000161  013440  000600  000000  B2TST:  NOP             @RDPF                   ;RESET DPF TO SEE IF BOARD 2 IS THERE.
000162  013440  060400  150035          NOP\F           @SDPF   %JDPF   ERRB2E  ;ERROR IF NO BOARD 2, ELSE
                                                                                ; SET DPF TO SEE IF BOARD 2 IS THERE.
000163  013440  060500  050035          NOP\F           @RSE    %JNDPF  ERRB2E  ;ERROR IF DPF NOT SET
000164  013440  000600  110524          NOP             @RDPF   %JMP    TSTRTN  ;TEST DER / RETURN

                                ;;*****************************************************************************
                                ;
                                ;LEV01 OR LEV02
                                ;       CLRERR
                                ;               CLEAR THE ERROR REG (R11)
                                ;
                                ;;*****************************************************************************

000165  034451  000011  127777  CLRERR: CLR     DER             %RET            ;CLEAR AND RETURN
```

```
                                ;;******************************************************************************
                                ;
                                ; TEST DESCRIPTION:
                                ;       The 2901 routine tests the internal 2901 registers, ALU functions,
                                ;       SHIFT/ROTATE functions and ALL ALU associated flags (ZERO, CARRY,
                                ;       NEGATIVE, LSB, OVERFLOW).
                                ;
                                ; Input:
                                ;       XNOR    R0,R0                           ;R0 = FFFF NEG/LSB SET
                                ;
                                ; Output:
                                ;       All registers are used in verifying register integrity &
                                ;       some registers are used in the ALU functionality.
                                ;
                                ; If any ERROR is detected within the 2901 test, the processor will report
                                ; immediately and stop any further execution.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104040 - ALU test error
                                ;
                                ;;******************************************************************************

000166  136540  040200  140031  ALUTST: XOR\F    #BIT15,R0       %JNLSB  ERRB1A  ;R0 = 7FFF NEG CLEARED
000167  035540  030001  140031          BIC\F    #BIT00,R0       %JNEG   ERRB1A  ;R0 = 7FFE LSB CLEARED
000170  004240  040000  050031          CLR\F    Q               %JLSB   ERRB1A  ;Q = 0 ZERO SET
000171  033441  010000  050031          MOV\F    R0,R1           %JNZRO  ERRB1A  ;R1 = 7FFE ZERO CLEARED
000172  020140  017001  140031          ADD\F    R1\0,R0,BAR     %JZRO   ERRB1A  ;AD = 2/R0 = FFFC/NEG SET
000173  100240  030000  040031          INC\F    Q               %JNNEG  ERRB1A  ;Q = 1 CARRY CLEARED
000174  102240  020000  150031          NEG\F    Q               %JCRY   ERRB1A  ;Q = FFFF LSB SET
000175  000041  040001  140031          ADD\F    R1\0,Q          %JNLSB  ERRB1A  ;Q = 7FFD CARRY SET
000176  007240  020000  050031          COM\F    Q               %JNCRY  ERRB1A  ;Q = 8002 NEG SET
000177  004260  030000  050200          CLR\T    Q               %TNNEG          ;DO NOT EXECUTE AN INSTRUCTION/SIGNAL CHANGE?
000200  101640  030002  050031          SUB\F    #2,Q            %JNNEG  ERRB1A  ;Q = 8000 LSB CLEARED
000201  033240  040000  040031          MOV\F    Q,R0            %JLSB   ERRB1A  ;R0 = 8000 ZERO CLEARED
000202  116540  010200  150031          XOR\F    #BIT15,R0\N     %JZRO   ERRB1A  ;R0 = BIT15 ZERO SET
000203  033741  010017  050031          MOV\F    #15.,R1         %JNZRO  ERRB1A  ;COUNTER R1 = 15.
000204  043440  010000  140206  1$:     SHF\RQF R0               %JZRO   2$      ;SHIFT RIGHT WITH Q, R0...IF ZERO FLAG
                                                                                ;SET HERE...GO CHECK R0 AND Q FOR SWAP
000205  031441  000001  110204          DEC      R1              %JMP    1$      ;IF NOT DONE DECREMENT SHIFT COUNTER
000206  013440  000000  000000  2$:     TST      R0                              ;IS LSB SET IN R0?
000207  013240  040000  140031          TST\F    Q               %JNLSB  ERRB1A  ;ERROR IF NOT
                                                                                ;IS LSB SET IN Q?
000210  043440  040000  140031          SHF\RQF R0               %JNLSB  ERRB1A  ;ERROR IF NOT, ELSE
                                                                                ; SHIFT R0 & Q RIGHT ONCE MORE
                                                                                ;SET HERE...GO CHECK R0 AND Q FOR SWAP
000211  013440  000000  000000          TST      R0                              ;IS MSB SET IN R0?
000212  114640  030200  040031          BIT\F    #BIT15,Q        %JNMSB  ERRB1A  ;ERROR IF NOT
                                                                                ; IS MSB SET IN Q?
000213  003740  030017  140031          MOV\F    #15.,Q          %JMSB   ERRB1A  ;ERROR IF YES
```



```
                                ;;******************************************************************************
                                ;
                                ;       If Q NOT 0, then hang otherwise load Q with 15., for checking rest of
                                ;       registers.  Loop to check all registers by WALKING a bit from carry to
                                ;       carry with a register and writing and comparing it with the rest of
                                ;       the registers.  Any stuck bits will propagate and be detected.
                                ;
                                ;       R0 should be 100000 now
                                ;       This loop should start with R0 equal to 100000
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104040 - ALU test error
                                ;
                                ;;******************************************************************************

000214  073440  000000  000000  SFT20:  ROT\L    R0                              ;ROTATE R0 LEFT WITH BIT 15->BIT 0
000215  033441  000000  000000          MOV      R0,R1                           ;R1=R0
000216  033442  000001  010000          MOV      R1,R2                           ;R2=R1
000217  033443  000002  000000          MOV      R2,R3                           ;R3=R2
000220  033444  000003  000000          MOV      R3,R4                           ;R4=R3
000221  033445  000004  000000          MOV      R4,R5                           ;R5=R4
000222  033446  000005  010000          MOV      R5,R6                           ;R6=R5
000223  033457  000006  010000          MOV      R6,R17                          ;R17=R6
000224  033450  000017  000000          MOV      R17,R10                         ;R10=R17
000225  033451  000010  000000          MOV      R10,R11                         ;R11=R10
000226  033452  000011  010000          MOV      R11,R12                         ;R12=R11
000227  033453  000012  000000          MOV      R12,R13                         ;R13=R12
000230  033454  000013  000000          MOV      R13,R14                         ;R14=R13
000231  033455  000014  000000          MOV      R14,R15                         ;R15=R14
000232  033456  000015  010000          MOV      R15,R16                         ;R16=R15
000233  033447  000016  010000          MOV      R16,R7                          ;R17=R16...LINKS ALL REGISTERS
000234  036147  000000  000000          XOR      R0,R7                           ;HERE COMPARE R0 (IMAGE) WITH LAST REGISTER
                                                                                ;SUPPOSEDLY CONTAINING THE SAME INFO AS R0
000235  001240  010000  050031          DEC\F    Q               %JNZRO  ERRB1A  ;ERROR IF NOT EQUAL, ELSE
                                                                                ; DECREMENT SHIFT COUNTER.
000236  053440  010000  040214          ROT\RF   R0              %JNZRO  SFT20   ;LOOP IF Q NOT ZERO, ELSE
                                                                                ; CHECK TO SEE NEG ISN'T SET.
000237  133744  030362  100031          MOV      #<LEDS+INDIAG>,CRI %JNEG ERRB1A ;ERROR IF STILL NEG AFTER RIGHT ROTATE /
                                                                                ; ALL LEDS OFF & SET DIAG MODE.
```

```
                                ;;*****************************************************************************
                                ;
                                ;       FORCE THE PAR (pipeline address register) INTO THE VECTOR TABLE
                                ;
                                ;       R11 IS THE DER.  AS INPUT INTO THE VECTOR TABLE, IT EQUALS THE
                                ;       RAM PARITY ERROR SHIFTED VALUE.  ON RETURN IT SHOULD EQUAL THE RAM
                                ;       PARITY ERROR.
                                ;       PAR WILL HOLD THE ADDRESS IN THE VECTOR TABLE WHERE THE PROC WILL
                                ;       GO TO.  CRI IS TESTED IN THE VECTOR.  OPM BIT HAS TO BE CLEARED.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104040 - ALU test error
                                ;
                                ;;*****************************************************************************

000240  173751  000040  130242  PARTST: MOV\L   #ER.SAP,R11     %CALL   1$      ;TEST THE PAR / SET RETURN IN STACK
000241  036551  000004  100243          XOR     #ER.RAP,R11     %JMP    2$      ;TEST RESULTS OF R11 DURING PAR VECTOR.

000374                          HI.OFF  = HIPAR - 7400
000242  013740  002374  110032  1$:     MOV     #HI.OFF,\N,PAR  %JMP    ERRB1E  ;LOAD PAR OFFSET TO VECTOR TO HI ADDRESS /
                                                                                ; ERROR IF PAR NOT FORCED TO HI VECTOR ADR.
000243                                  ASSUME HI.OFF,EQ,374

000243  013440  010000  000032  2$:     NOP                     %JNZRO  ERRB1E  ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES
000244  173751  000040  120246          MOV\L   #ER.SAP,R11     %CALL   3$      ;TEST THE PAR-SET RETURN IN STACK
000245  036551  000004  110247          XOR     #ER.RAP,R11     %JMP    4$      ;TEST RESULTS OF R11 DURING PAR VECTOR.

000003                          LO.OFF  = LOPAR - 7400
000246  013740  002003  110032  3$:     MOV     #LO.OFF,\N,PAR  %JMP    ERRB1E  ;LOAD PAR OFFSET TO VECTOR TO LO ADDRESS /
                                                                                ; ERROR IF PAR NOT FORCED TO LO VECTOR ADR.
000247                                  ASSUME LO.OFF,EQ,3

000247  013440  010000  000032  4$:     NOP                     %JNZRO  ERRB1E  ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES

                                ;;*****************************************************************************
                                ;
                                ;       INCREMENT TEST (TSTCNT= Q) AND RETURN
                                ;
                                ;;*****************************************************************************

000250  100240  000000  110524  INCRTN: INC      TSTCNT          %JMP    TSTRTN  ;INCREMENT TSTCNT(Q) & RETURN
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEVO1
                                ;       CONTROL REGISTER DATA INTEGRITY TEST
                                ;
                                ;       Loop data around the D-PROC Control Register.
                                ;
                                ;       Don't test BIT05, this bit is not loadable from this register.
                                ;       Don't test BIT06, this bit is the enable RAM PE high and will
                                ;       be tested later.
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               none
                                ;
                                ;       REGISTERS USED:
                                ;               R1 is a temporary register.
                                ;
                                ;       This code starts at bit position BIT00,
                                ;       skips over bit position BIT05 (SERIAL),
                                ;       skips over bit position BIT06 (RAMPE),
                                ;       and ends at bit position BIT15 (LED8).
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105102 - Control register test error
                                ;
                                ;;*****************************************************************************

000251  033741  000001  010000  DCRTST: MOV     #BIT00,R1                       ;LOAD R1 WITH STARTING POINT
000252  013440  004001  000000  1$:     MOV     R1,DCRD                         ;LOAD CONTROL REGISTER
000253  033702  000001  000000          MOV     (DCRS),R2                       ;GET DATA FROM CONTROL REG
000254  035542  000040  000000          BIC     #SERIAL,R2                      ;IGNORE LO BYTE
000255  016141  000002  000000          XOR     R2,R1\N                         ;HAS DATA BEEN DESTROYED?
000256  073441  010001  040035          ROT\LF  R1              %JNZRO  ERRB2E  ;ERROR IF SO, ELSE SHIFT DATA.
000257  016541  030040  150524          XOR\F   #SERIAL,R1\N    %JMSB   TSTRTN  ;EXIT IF LAST BIT TESTED, ELSE
                                                                                ; IS THIS THE SERIAL BIT?
000260  073461  010001  140261          ROT\LT  R1              %TZRO           ;ROTATE LEFT IF YES, ELSE
                                                                                ; CONTINUE.
000261  016541  000100  010000          XOR     #RAMPE,R1\N                     ;IS THIS THE RAMPE BIT?
000262  073461  010001  150263          ROT\LT  R1              %TZRO           ;ROTATE LEFT IF YES, ELSE
                                                                                ; CONTINUE.
000263  013440  000000  110252          NOP                     %JMP    1$      ;DO NEXT BIT POSITION
```

```
                                ;;*****************************************************************************
                                ;
                                ;       TEST ROM PARITY ERROR
                                ;
                                ;       Input:
                                ;               STACK set for return.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104041 - ROM parity test error
                                ;
                                ;;*****************************************************************************

000264  013440  000000  130013  TSBRPE: NOP                     %CALL   TSCRPE  ;A CONTROL ROM PARITY ERROR
000265  013440  000000  100032          NOP                     %JMP    ERRB1E  ;ERROR IF IT DOESN'T VECTOR,
                                                                                ; SHOULD VECTOR BEFORE JUMPING TO ERROR ROUTINE.
```



```
                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;       D-PROC CROM PE & TIMEOUT TEST
                                ;
                                ;       REGISTERS USED: R11 (DER) HOLDS THE VECTOR ERROR CODE
                                ;
                                ;       CALLS   TSCRPE -> A KNOWN BAD PARITY LOCATION.
                                ;               TSTRTN - ONE INSTRUCTION (RET) FOR TIMING AND SETTING
                                ;                        THE STACK
                                ;               TMOUT TO TEST THE D-PROC'S TIMEOUT CAPABILITY
                                ;
                                ;       NOTE:
                                ;               THE MAXIMUM TIME ALLOWABLE FOR A TIME OUT IS
                                ;               .177778 SECONDS = 45HZ*8.   SO THE NUMBER OF
                                ;               TIMES THROUGH A NINE INSTRUCTION LOOP = 61220.
                                ;               .177778 = 9(# OF INST) * 363*10**-9(INST TIME) * 61220.
                                ;
                                ;               THE MINIMUM TIME ALLOWABLE FOR A TIME OUT IS
                                ;               .107778 SECONDS = 65HZ*7.   SO THE NUMBER OF
                                ;               TIMES THROUGH A NINE INSTRUCTION LOOP = 37114.
                                ;               .107778 = 8 * 363*10**-9 * 37114
                                ;
                                ;               THE DIFFERENCE BETWEEN THE TWO TIMES IS .07 SECONDS
                                ;               THE LOOP DIFFERENCE = 24106.  IF THE MINIMUM TIME OUT
                                ;               OCCURED, R12 SHOULD = 24106.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104041 - ROM parity test error
                                ;       104041 - Timeout test error
                                ;
                                ;;*****************************************************************************

177777                          MAXTIM  := 65535.
141520                          MINTIM  := 50000.

000266  034453  000013  120264  DROMPE: CLR     R13             %CALL   TSBRPE  ;TEST ROM PE / R13 = 0 FOR TIME OUT TEST
000267  036551  000005  000000          XOR     #ER.ROP,R11                     ;CHECK ERROR CODE IN D ERROR REG
000270  133752  010377  050032          MOV\F   #MAXTIM&HIBYT,R12 %JNZRO ERRB1E ;ERROR IF NOT ZERO, ELSE
                                                                                ; R12 = LOOP COUNTER
000271  033552  000377  120276          BIS     #MAXTIM&LOBYT,R12 %CALL TMOUT   ;GO TO A TIMEOUT
000272  036551  000015  010000          XOR     #ER.TMO,R11                     ;CHECK ERROR CODE IN D ERROR REG
000273  131153  010012  050032          SUB\F   R12,R13         %JNZRO  ERRB1E  ;ERROR IF NO VECTOR OCCURED, ELSE
                                                                                ; SEE IF TIME OUT TOO SOON.
000274  133744  034363  100032          MOV     #<LEDS+ODDP+INDIAG>,CRI,DCRD %JNEG ERRB1E ;ERROR IF TIMEOUT /
                                                                                        ; ALL LEDS+ODD PARITY+DIAG MODE.
000275  013440  000000  100250          NOP                     %JMP    INCRTN  ;JUMP TO INCREMENT TEST NUMBER AND
                                                                                ; RETURN.
```

```
                                ;;******************************************************************************
                                ;
                                ;LEVO2
                                ;       TMOUT
                                ;               TIMEOUT ROUTINE
                                ;
                                ;               The timeout circuit is enabled when ever a successful jump is
                                ;               executed.  Falling through to the next instruction resets the
                                ;               timeout circuit.  Timeout errors DO NOT occur on the UPROC.
                                ;
                                ;               This routine will loop until a timeout occurs or until the
                                ;               maximum timeout limit is exceeded.
                                ;
                                ;       REGISTER USED:
                                ;               R12 is used for loop control.
                                ;               R13 gets the value reflecting the minimum timeout allowed.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       104041 - Timeout test error
                                ;
                                ;;******************************************************************************

000276  133544  014362  010300  TMOUT:  BIS     #<LEDS+INDIAG>,CRI,DCRD %JNZRO 1$  ;IF NOT DONE, WASTE MORE TIME
000277  013440  000000  100032          NOP                     %JMP    ERRB1E     ;ERROR IF HERE.

000300  133553  000074  100301  1$:     BIS     #<MAXTIM-MINTIM>&HIBYT,R13 %JMP 2$ ;FOLLOWING LOOP FOR TIME OUT
000301  033553  000257  110302  2$:     BIS     #<MAXTIM-MINTIM>&LOBYT,R13 %JMP 3$ ; MAX TIME OUT = .17778 SEC = 45HZ*8
000302  013440  000000  110303  3$:     NOP                     %JMP    4$
000303  031452  000012  110276  4$:     DEC     R12             %JMP    TMOUT      ;LOOP ITERATION(61220-MAX OR 37116-MIN)

                                ;;******************************************************************************
                                ;
                                ;       END OF ALL BOARD 1 TESTING
                                ;
                                ;;******************************************************************************
```



```
                                ;;******************************************************************************
                                ;
                                ;LEVO1
                                ;       RAM PE
                                ;
                                ;       USES RO, R11(DER), BAR AND BUF LOC O & 1
                                ;
                                ;       CALLS   THE 'RAMPET' ROUTINE FORCES A RAM PARITY ERROR
                                ;               FOR ITSELF, THEN SETS UP A PARITY ERROR FOR THE
                                ;               U-PROC BY SETTING A PARITY ERROR IN ADDRESS O
                                ;               OF RAM. ITS PARITY ERROR IS IN ADDRESS 1 OF RAM.
                                ;
                                ;       RAM ADDRESS O CONTAINS THE U-PROC RAM PE
                                ;       RAM ADDRESS 1 CONTAINS THE D-PROC RAM PE
                                ;
                                ;       - Only the D-PROC can enable RAM PARITY errors -
                                ;
                                ;       Input:
                                ;               Set ODD PARITY (ODDP) in DCR.
                                ;       Output:
                                ;               R1 is a temporary reg.
                                ;               R11 (DER) is cleared.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105102 - RAM parity error
                                ;
                                ;;******************************************************************************

000304  133745  000100  130311  DRAMPE: MOV     #<1024.*16.>,UBAR %CALL RAMZAP  ;SET UBAR TO 16K, ZERO RAM WITH ODD PARITY SET
000305  013440  000000  130315          NOP                     %CALL  RAMPET  ;FORCE PARITY ERROR
000306  036551  000004  010000          XOR     #ER.RAP,R11                    ;TEST RESULTS OF R11 DURING RAM PARITY VECTOR.
000307  013440  010000  010035          NOP                     %JNZRO ERRB2E  ;ERROR IF INCORRECT RESULTS DURING
                                                                               ; VECTOR ROUTINES.
000310  033444  004004  100250          MOV     CRI,,DCRD       %JMP   INCRTN  ;RESET LEDS & REENABLE SIGNALS /
                                                                               ; INCREMENT TSTCNT(Q) & RETURN.

                                ;;******************************************************************************
                                ;
                                ;       ROUTINE TO SAVE LAST FAILURE AND ZAP RAM
                                ;
                                ;;******************************************************************************

000311  031445  007005  010000  RAMZAP: DEC     UBAR,,BAR                      ;LOAD RAM ADDRESS /
                                                                               ; DONE ZEROING RAM?
000312  016545  030034  127777          XOR     #FAILUR,UBAR\N  %RNEG          ;RETURN IF DONE /
                                                                               ; IS UBAR = FAILUR?
000313  034442  010002  173602          CLR\F   R2              %CZRO  S.LDR2  ;SAVE BUFFER DATA IN R2, ELSE
                                                                               ; CLEAR R2.
000314  013440  003002  100311          MOV     R2,BUF          %JMP   RAMZAP  ;LOAD R2 INTO BUFFER / NEXT RAM LOCATION
```

```
                                ;;*****************************************************************************
                                ;
                                ;LEV02
                                ;       SET UP THE RAM PARITY ERROR IN LOC 0 AND 1
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105102 - RAM parity error
                                ;
                                ;;*****************************************************************************

000315  133744  004362  010000  RAMPET: MOV     #<LEDS+INDIAG>,CRI,DCRD         ;SET EVEN RAM PARITY.
000316  014440  007000  133623          CLR     ,\N,BAR         %CALL   S.CLRB  ;LOC 0 <- PARITY ERROR, WRITE EVEN PARITY
000317  133544  000001  123623          BIS     #ODDP,CRI       %CALL   S.CLRB  ;LOC 1 <- PARITY ERROR, WRITE EVEN PARITY /
                                                                                ; SET ODD PARITY IN CRI.
000320  033544  004100  010000          BIS     #RAMPE,CRI,DCRD                 ;ENABLE RAM PE.
000321  013740  007001  110035          MOV     #1,\N,BAR       %JMP    ERRB2E  ;CAUSE PROC TO VECTOR BY LOADING
                                                                                ; THE DBAR -> ADRS 1 WITH ODD PARITY.
                                                                                ; VECTOR WILL OCCUR AFTER THIS INSTRUCTION,
                                                                                ; IF NOT, REPORT ERROR.
```



```
                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;       RAM BUFFER TEST (Duration 3.3 sec)
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               Each location in RAM will be ZERO.
                                ;
                                ;       REGISTERS USED:
                                ;               R0 has OLD buffer value
                                ;               R1 has NEW buffer value
                                ;               R5(UBAR) has BAR value
                                ;               R6 is the increment value (1 or -1)
                                ;               R10 saves the starting value
                                ;
                                ;       TEST DESCRIPTION:
                                ;
                                ;             The Algorithm;
                                ;
                                ;             1.  Fill the RAM with all zeros.
                                ;
                                ;             2.  Starting with the lowest address,  read  a  location,
                                ;                 verifying the previous write, write a one in a single
                                ;                 bit position (starting at the LSB), and read the same
                                ;                 location, verifying the write.
                                ;
                                ;             3.  Repeat step 2  until the  highest address in the  RAM
                                ;                 is reached.
                                ;
                                ;             4.  Starting again at the lowest address, repeat steps  2
                                ;                 and 3 for every bit of the data word.
                                ;
                                ;             5.  Starting again at the lowest address, repeat steps 2,
                                ;                 3,  and 4, but this time putting zeros back into each
                                ;                 bit.
                                ;
                                ;             6.  Now, starting with the highest address, repeat  steps
                                ;                 2,  3,  4,  and 5, but stepping through the memory in
                                ;                 reverse  order,  from  highest  address  to  lowest
                                ;                 address.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105105 - RAM buffer error
                                ;
                                ;;*****************************************************************************
```

```
000322  033746  000001  130250  RAMTST: MOV     #1,R6             %CALL   INCRTN  ;INITIALIZE R6 VALUE /
                                                                                  ; INCREMENT TSTCNT(Q) & RETURN.
000323  133745  000100  130311          MOV     #<1024.*16.>,UBAR %CALL RAMZAP  ;GO INITIALIZE RAM

000324  034440  000000  000000          CLR     R0                              ;INITIALIZE OLD BUFFER VALUE
000325  133750  000100  100343          MOV     #<1024.*16.>,R10  %JMP  4$      ;INITIALIZE R10 VALUE (16K)

000326  016542  000034  010000  1$:     XOR     #FAILUR,R2\N                    ;DOES UBAR = FAILUR?
000327  033445  017005  140334          MOV\F   UBAR,,BAR         %JZRO   2$    ;JUMP IF IT DOES, ELSE
                                                                                  ; IS UBAR NEGATIVE?
000330  016500  030003  100337          XOR     (BUF),R0\N        %JNEG   3$    ;EXIT LOOP IF DONE /
                                                                                  ; IS OLD BUFFER VALUE OK?
000331  013441  013001  040035          MOV\F   R1,\N,BUF         %JNZRO  ERRB2E  ;ERROR IF NOT, ELSE
                                                                                  ; WRITE NEW VALUE TO BUFFER.
000332  033445  007005  000000          MOV     UBAR,,BAR                       ;RESET RAM ADDRESS
000333  016501  000003  010000          XOR     (BUF),R1\N                      ;IS NEW BUFFER VALUE OK?
000334  030145  010006  050035  2$:     ADD\F   R6,UBAR           %JNZRO  ERRB2E  ;ERROR IF NOT, ELSE
                                                                                  ; SET NEXT RAM ADDRESS.
000335  033442  000005  000000          MOV     UBAR,R2                         ;SAVE RAM ADDRESS IN R2
000336  135542  000300  100326          BIC     #BIT15!BIT14,R2 %JMP    1$      ;GET RID OF JUNK BITS &
                                                                                  ; DO NEXT COMPARE.
000337  033440  000001  000000  3$:     MOV     R1,R0                           ;THE NEW BUFFER VALUE IS NOW THE OLD BUFFER VALUE
000340  176541  010200  010344          XOR\L   #100000,R1        %JNZRO  5$    ;COMPUTE NEXT NEW BUFFER VALUE
000341  132446  000006  010000          NEG     R6                              ;CHANGE SENSE OF R6
                                                                                  ; DONE WITH TEST (DID R6 GO FROM -1 TO 1) ?
000342  137750  030300  010250          MOV     #<1024.*16.>-1,R10 %JNNEG INCRTN ;RETURN IF DONE /
                                                                                  ; R10 HAS NEW ENDING ADDRESS (16k-1).
000343  033741  000001  010000  4$:     MOV     #1,R1                           ;INITIALIZE NEW BUFFER VALUE
000344  033445  007010  110326  5$:     MOV     R10,UBAR,BAR      %JMP    1$    ;RESET STARTING VALUE
```



```
                                ;;*****************************************************************************
                                ;
                                ;LEVO2
                                ;       SDI CLOCK ROUTINE
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105152 - SDI error
                                ;
                                ;;*****************************************************************************

000345  034515  110002  000036  SDICLK: AND     (RTDS),R15        %JNCSR  ERRB2H  ;ERROR IF RTCS NOT READY
000346  016155  150014  000036          XOR     R14,R15\N         %JNDSR  ERRB2H  ;ERROR IF RTDS NOT READY
000347  013440  010000  010036          NOP                       %JNZRO  ERRB2H  ;ERROR IF RTDS DATA NOT RECEIVED AS SENT BY RTCS

000350  170456  145016  150036  SDICL1: INC\LF  R16,,RTCS         %JDSER  ERRB2H  ;ERROR IF RTDS PULSE OR PARITY ERROR
                                                                                  ; RTCS= 1,3,7,F,1F,3F,7F,FF

000351  013440  110000  140036  SDICL2: NOP\F                     %JCSR   ERRB2H  ;ERROR IF RTCS NOT CLEARED BY RTCS LOAD
000352  133755  150200  100036          MOV     #RWRDY,R15        %JDSR   ERRB2H  ;ERROR IF RTDS NOT CLEARED BY RTCS LOAD
                                                                                  ;
000353  033555  000163  100507          BIS     #<ATTN+DRDY+SEC+IDX+AVAIL>,R15 %JMP OCLK48 ;SEND NEXT RTCS FRAME.
                                                                                  ; R15= RTCS MASK
```

```
                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;       SDI INTERCONNECT TEST
                                ;
                                ;       This test sends data to the diagnostic wrap port by clocking it
                                ;       x amount of times, then checks for the proper signals.  All SDI
                                ;       related signals are checked for stuck at high  or  low  states.
                                ;       Data is clocked thru the TTL (32. clocks) and ECL (33. clocks),
                                ;       then verified.
                                ;
                                ; NOTE: If a port wrap connector is inserted in J3 on the SDI board and
                                ;       PIN 23 (EXT.TEST L) is grounded,  then all 4 ports (3 - 0) will
                                ;       be tested in  the same manor  as the diagnostic wrap port. This
                                ;       ensure that the SI RCV/DRV components will  be tested  for each
                                ;       selected port.
                                ;
                                ;       REGISTERS USED:
                                ;               R11 is NOT USED
                                ;               R12 has CLOK16 shift data to serdes read input
                                ;               R13 has CLOK16 number of clocks
                                ;               R14 has RTDS compare data
                                ;               R15 has RTDS bit mask
                                ;               R16 has RTCS data
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105152 - SDI error
                                ;
                                ;;*****************************************************************************

000354  135564  064360  140355  SDITST: BIC\T   #LEDS,CRI,DCRD  %TDPF           ;IF DPF IS SET, REMEMBER WE ARE DOING
                                                                                ; LOOP-ON-MICROTEST

                                        ;;Get RTDS in sync with DPROC - RTCS & RTDS will always be at an unknown sync

000355  034456  005476  130506          CLR     R16,,RTCS @SCLR %CALL OCLKRS    ;CLEAR OUT RTCS / SET CLEAR / DO MANY CLOCKS
000356  033756  005001  120507          MOV     #1,R16,RTCS     %CALL   OCLK48  ;SET CONTINOUS & SEND FRAME WITH CORRECT PARITY
000357  073456  005016  000000          ROT\L   R16,,RTCS

                                        ;;Sync RTDS to DPROC

000360  013440  150000  120510          NOP                     %CDSR   OCLK1   ;CLOCK RTDS ONCE IF RTCS LOAD DID NOT CLEAR %DSR
000361  034454  000674  120350          CLR     R14       @RCLR %CALL   SDICL1  ;R14= RTDS COMPARE DATA / GET FIRST FRAME IN RTDS /
                                                                                ; RESET I/O CLEAR.

                                                                                ;Send:                  Test for on next call:
                                                                                ;-----                  ----------------------
000362  033554  000002  120345          BIS     #ATTN,R14       %CALL   SDICLK  ;RTCS= #WRT,            RTDS= #ATTN
000363  133554  000200  130345          BIS     #RWRDY,R14      %CALL   SDICLK  ;RTCS= RTCS+#RD,        RTDS= RTDS+#RWRDY
000364  033554  000001  120345          BIS     #DRDY,R14       %CALL   SDICLK  ;RTCS= RTCS+#CRDY,      RTDS= RTDS+#DRDY

                                        ;;Send 1 WORD of ZERO'S - RTDS Stays the same

000365  031456  000016  120350          DEC     R16             %CALL   SDICL1  ;SEND NEXT DATA FRAME IF #CONT FAILURE= #SEC+#IDX
```



```
000366  053456  000016  010000          ROT\R   R16


                                        ;;These bits match between RTCS & RTDS

                                                                                ;Send:                  Test for on next call:
                                                                                ;-----                  ----------------------
000367  033554  000020  120345          BIS     #SEC,R14        %CALL   SDICLK  ;RTCS= RTCS+#INI,      RTDS= RTDS+#SEC
000370  033554  000040  120345          BIS     #IDX,R14        %CALL   SDICLK  ;RTCS= RTCS+#RTCS13,    RTDS= RTDS+#IDX
000371  033554  000100  120345          BIS     #AVAIL,R14      %CALL   SDICLK  ;RTCS= RTCS+#RTCS14,    RTDS= RTDS+#AVAIL
                                                                                ; PARITY ERROR WILL BE GENERATED AFTER CLOCK CALL

                                        ;;Force PARITY ERROR

000372  034456  140476  040036          CLR\F   R16     @SCLR   %JNDSER ERRB2H  ;ERROR IF NO RTDS PARITY ERROR
000373  034452  000672  130345          CLR     R12     @RCLR   %CALL   SDICLK  ;SEND CORRECT PARITY TO STOP FUTURE ERRORS /
                                                                                ; RESET I/O CLEAR.

                                        ;;SDI test done

                                ;
                                ;       RTCS DATA  = RTDS DATA= ????, ????, 0101, 01
                                ;       WRITE DATA = READ DATA= ????, 0000, 0001, 0003, 0007, 000F, 001F, 003F, 007F, 00FF, 01FF
                                ;
```

```
;;*****************************************************************************
;
;LEVO1
;       SDWRT:  SERDES WRITE MODE TEST
;       SDRD:   SERDES READ  MODE TEST
;
;       FOR THE SERDES WRITE TEST:
;
;NRZ SERIAL DATA IN >-+---------(1)-----------+
;                     !         +----------+   !
;                   +---!---------!---+      !   !
;                   V   V         !   !      V   V
;                  +WRT--RD+      !   !    +WRT--RD+          +--------+          +--------+
;    ----(16)--->!        !-->+   !   !    !        !<-+--(10)->!        !--+-(10)->!        !
;       26BCH,   ! SERDES!   !   !   !    ! SERDES!   !        ! RSGEN !  !        !   ECC  !
;       FFFFH    !   16  !   !   !   !    !   10  !   !        !       !  !        !        !
;                !       !   !   !   !    !       !   !        !     ERR-----(1)->!        !
;                +TS--TTL+   !   !   !    +TS--TTL+   !        +--------+   !    +--------+
;                   !   !    !   !   !       !        +-----------------<-+
;                   V   +-(1)-!---+   !       V
;                   +---------!--------+-(1)-+---> NRZ SERIAL DATA OUT
;                             (16)
;                              !
;                              V
;                         +--------+
;                         !  ---   !
;                         !  \      !  SOFTWARE CALCULATES THE SUM
;                         !  /      !
;                         !  ---   !
;                         +--------+
;
;      @SRM    @SCMD     WHAT HAPPENS IF @SSE & @SECC
;      ----    -----     ---------------------------
;       0       0        WRITE MODE - TTL WRAP SD16 -> SD10 ->SD16 & NRZ OUT
;       0       1        WRITE MODE - TTL WRAP SD16 -> SD16 & NRZ OUT
;       1       0        READ  MODE - TTL WRAP NRZ DATA IN -> SD16 & SD10
;       1       1        READ  MODE - TTL WRAP NRZ DATA IN -> SD16 & SD10 -> NRZ OUT
;
; NOTE: The Diagnostic Clock and data out can be injected at the ECL encoder
;       then wrapped through to the ECL decoder to the NRZ Serial data in. It
;       can also be wrapped bypassing the ECL and only using TTL paths. There
;       is a one clock delay through the ECL path.
;
;;*****************************************************************************
```



```
;;*****************************************************************************
;
;       Input:
;               none
;       Output:
;               ECC output will change.
;               REGS R7-R16 "    "
;
;       REGISTERS USED:
;               R11 = USED FOR LOOP COUNTER
;               R12 = HOLDS SERIAL DATA
;               R13 = HOLDS # OF CLOCKS
;               R14 = TEMP & HOLDS ECC REGISTER VALUE
;               R15 = (DBAR) FOR BAR INCREMENT
;               R16 = HOLDS ECC SUM
;               R17 = SERDES PARALLEL DATA. (READ)= EDC, (WRITE)= WRITE DATA,
;               LOAD R17 WITH VALUE - 1
;
;       CALLS   SD.RS, CLK, SDCLK, GENSYN, GETECC, D.GECC
;
;;*****************************************************************************
```

```
                                ;;*****************************************************************************
                                ;
                                ;       WRITE MODE - WRAP SERDES 16 ONLY
                                ;
                                ;       This test checks the parallel data, residue ready,  overrun and
                                ;       WRC  signals  for  being  stuck high or low.  A sync pattern is
                                ;       loaded in the SERDES 16.  After the required amount of  clocks,
                                ;       a  pattern  is  read.   The  pattern  has become skewed by this
                                ;       method, but is equal to  a  specific  value.   The  pattern  is
                                ;       verified and checked for Stuck at high or low states.
                                ;
                                ;      WORD          1     2     3     4                          254  255 256  257      258 ->
                                ;WRITE DATA = 26BC, 8000, C000, E000, F000, ... FFFF, 0, 8000, ... FFFF, 0, 8000, EDC= C000, ECC
                                ;
                                ; - = 2 BIT TIMES          WRC RELATIONSHIP IF SERDES 16 IN WRONG MODE
                                ;
                                ; 8 BIT --  --  --  --  --  -
                                ; SD16    --  --  --  --  --
                                ;                      !!
                                ;10 BIT --   --   --   --   -
                                ; SD16    ---  ---  ---  ---
                                ;                      !! TEST WRAP HERE & HERE
                                ;16 BIT ----    ----    ----    ----    ----    ----
                                ; SD16      ----    ----    ----    ----    ----
                                ;                       !@SECCT          !! TEST %RRDY
                                ;10 BIT ------------------  --   --   --   --
                                ; SD10                     ---  ---  ---  ---
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105153 - Write mode, wrap SERDES 16 error
                                ;
                                ;;*****************************************************************************

000374  033751  000050  130514  SDWRT:  MOV     #<39.+1>,R11    %CALL   SD.RST  ;CLOCKS TO FIRST TEST POINT. INITIALIZE SERDES

                                        ;;Send PARALLEL data

000375  033453  000711  130533          MOV     R11,R13 @SSE    %CALL   GENSYN  ;SET SERDES ENABLE /
                                                                                ; GENERATE SYNC PATTERN & SERDES
000376  014440  005740  010000          CLR     R0,RTCS @SECC                   ;SET ECC ENABLE (SERDES 10 ENABLE) FOR RRDY TEST


                                        ;;Send the SYNC character

000377  024452  006452  120513          CLR     R12\0,R12,SD @SCMD %CALL OCLK00 ;SEND SYNC PATTERN TO SERDES & RETURN FROM CLOCK
                                                                                ; ROUTINE JUST BEFORE WRC DOWN THIRD TIME
000400  133754  170257  110036          MOV     ##257,R14       %JWRC   ERRB2H  ;ERROR IF WRC TO EARLY /
                                                                                ; R14 = SHIFTED SYNC= D780H
000401  150454  000774  120510          INC\R   R14     @SECCT  %CALL   OCLK1   ;CLOCK 1 TIME / SET ECC TIME
                                                                                ; ALLOWS FOR ECL OR TTL CLOCK WRAP
000402  013440  160720  010036          NOP             @SRM    %JNOVER ERRB2H  ;ERROR IF OVERRUN NOT TURNED ON
000403  016514  170646  010036          XOR     (SD),R14\N @RCMD %JNWRC ERRB2H  ;ERROR IF WRC LATE /
                                                                                ; SD= D780H= 153600 Q
                                ;temp   bic     #bit15,cri      %jnzro  errb2h  ;ERROR IF NOT WRAP PROPERLY. R16= ECC SUM
```



```
000404  013440  010600  010036          NOP             @RDPF   %JNZRO  ERRB2H  ;ERROR IF NOT WRAP PROPERLY. R16= ECC SUM
                                                                                ; WRITE MODE - @RDPF= WRITE CLKS

                                        ;;Test %RRDY

000405  033753  000045  130511          MOV     #37.,R13        %CALL   OCLK    ;GENERATE 37. CLOCKS
000406  033755  007060  120525          MOV     #60,DBAR,BAR    %CALL   D.RRDY  ;TEST %RRDY / DBAR, BAR -> LOC 60
```

```
                                ;;*****************************************************************************
                                ;
                                ;       WRITE MODE - WRAP SERDES 16 & SERDES 10
                                ;
                                ;;*****************************************************************************

000407  033751  000021  130514            MOV     #17.,R11        %CALL   SD.RST  ;RESET IOC'S/R11 HAS CLOCK COUNT

                                ;;Send PARALLEL data

000410  033453  000711  130533            MOV     R11,R13 @SSE    %CALL   GENSYN  ;GENERATE SYNC PATTERN & SERDES

                                ;;Send the SYNC character

000411  024452  006752  120463  TP010:    CLR     R12\0,R12,SD @SECC %CALL CLKP   ;SEND SYNC PATTERN TO SERDES CLOCK 17 TIMES SO
                                                                                ; RETURN FROM CLOCK ROUTINE AT START OF WORD.
                                                                                ; CLEAR R17 SO DATA= 1,3,7, ... FFFF, 0, 1 ...
                                                                                ; READ DATA = 1,3,7 ???

                                ;;Self adjustment for TTL or ECL wrap

000412  033750  100060  020462            MOV     #60,R10         %CNPRDY CLK1    ;ADJUST FOR ECL OR TTL WRAP.

                                ;;Send 256. WORDS of data (10. BITS at a time)

000413  033751  000005  120501            MOV     #5.,R11         %CALL   SDCLK   ;WORDS 1- 4. GET TO WHERE ECC IS VALID
000414  034456  100016  010036            CLR     R16             %JNPRDY ERRB2H  ;ERROR IF 4TH PRDY NOT SET. (IN WORD 4 BEFORE LOAD)
000415  073751  000312  130501            MOV\L   #<404./2>,R11   %CALL   SDCLK   ;WORDS 5-256

                                ;;Load WORD 257. - 258.

000416  033751  000004  130501            MOV     #4.,R11         %CALL   SDCLK   ;WORD 257-258. R11= 4 10 BIT WORDS FOR @RECCT
                                                                                ; LET WORD 257= C000H. IT IS NORMALLY THE EDC WORD

                                ;;Load word 259. and set ECC time enable (@SECCT)

000417  033455  007770  010000  TP011:    MOV     R10,DBAR,BAR @SECCT             ;ECC1 IN BAR= 62. HERE IN WORD 259 BEFORE LOAD
000420  033751  000020  130501            MOV     #16.,R11        %CALL   SDCLK   ;WORDS 259-268

                                ;;Load word 269. and reset ECC time enable (@RECCT)

000421  033751  000010  000000            MOV     #8.,R11                         ;R11 = 273-269 WORDS IN 10 BIT MODE
000422  034457  000577  130501            CLR     R17     @RECCT  %CALL   SDCLK   ;WORDS 269-274. HERE IN WORD 274 BEFORE LOAD
                                                                                ; R17= 0 SO WRAP DATA = 3000H

                                ;;Load WORD 274.

000423  133753  000060  120531            MOV     ##60,R13        %CALL   S.SRM   ;SET READ MODE AND XOR (SD),R11\N @RECC
                                                                                ; HERE IN WORD 274 BEFORE LOAD
000424  013440  010520  000036            NOP             @RRM    %JNZRO  ERRB2H  ;ERROR IF @RECCT DID NOT WORK PROPERLY.
                                                                                ; WRAP DATA WRONG BECAUSE @RCMD NOT WRAPPING
                                                                                ; SD16 -> SD10 -> SD16

                                ;;Load WORD ???.
```



```
000425  033753  000040  130464            MOV     #32.,R13        %CALL   CLK     ;CLK16-> 1 CLK32-> 0
000426  133751  000337  120531            MOV     ##337,R11       %CALL   S.SRM   ;R13= 0. R17= EDC VALUE
000427  030156  010011  010036            ADD     R11,R16         %JNZRO  ERRB2H  ;ERROR IF @RECC NOT WORKING.
                                                                                ; DATA SHOULD BE = 0
                                                                                ; R16 ADDITION TO MAKE ZERO FOR TEST
```

```
                                ;;*****************************************************************************
                                ;
                                ;       READ MODE - SERIAL LOAD SERDES 16 & SERDES 10
                                ;
                                ; NOTE: This uses 4 LEVELS so may have to be in calling level
                                ;
                                ;;*****************************************************************************

000430  030556  000000  130440  SDRD:   ADD     #0,R16          %CALL   ECCRD   ;MAKE R16= 0 FOR LAST TEST
000431  033755  037060  010036          MOV     #60,DBAR,BAR    %JNNEG  ERRB2H  ;ERROR IF RSGEN GIVES ECC ERROR
                                                                        ; CLEAR DBAR SO ECCRD: WILL WRITE OVER ECC SYMBOLS
                                TP014:
000432  030156  000017  120440  TP016:  ADD     R17,R16         %CALL   ECCRD   ;TEST FOR ECC ERROR. ADD READ EDC TO ECC SUM
000433  030156  030017  110036          ADD     R17,R16         %JNEG   ERRB2H  ;ERROR IF RSGEN NOT GIVE ECC ERROR
                                                                        ; ADD READ EDC TO ECC SUM
000434  033751  000042  120537          MOV     #<17.*2>,R11    %CALL   GETECC  ;GET ECC RESIDUES= ECC ERROR
000435  013444  010004  040036          TST\F   CRI             %JNZRO  ERRB2H  ;ERROR IF ECC SUM (R16) NOT CORRECT

000436  013460  030600  150437          NOP\T           @RDPF   %TNEG           ;RESET DPF IF NO LOOP-ON-TEST

000437  133544  004360  110250          BIS     #LEDS,CRI,DCRD  %JMP    INCRTN  ;DESELECT DIAGNOSTIC WRAP PORT
```



```
                                ;;*****************************************************************************
                                ;
                                ;       READ MODE - TEST SERDES 16, SERDES 10, RSGEN AND ECC CIRCUITRY
                                ;
                                ;       This test simulates writing a sector's worth of data to the disk
                                ;       and makes sure that the timing between the SERDES 16 and SERDES
                                ;       10 is correct and the summation of  wrapped  data  doesn't  get
                                ;       altered.   Then,  the  test  simulates  a read from a disk.  It
                                ;       tests the ECC circuitry and the RSGEN chip.  An error indicator
                                ;       on the RSGEN is tested for being stuck high or low.  The timing
                                ;       between the SERDES 16, SERDES 10 and the RSGEN is checked.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105154 - Read mode, SERDES 16, 10 RSGEN and ECC circuitry error
                                ;
                                ;;*****************************************************************************

                                        ;;Entry condition, R13 = 0

000440  013440  000000  130516  ECCRD:  NOP                     %CALL   SD.RS   ;RESET IOC'S & DO NOT CLOCK. R13= CLOCK COUNT

                                        ;;Send PARALLEL data

000441  013456  000716  130533          TST     R16     @SSE    %CALL   GENSYN  ;GENERATE SYNC PATTERN & SERDES
                                                                        ; ADD R17 TO R16 SO R16 WILL KEEP ECC AND EDC SUM
                                                                        ; TST R16 is for DEBUG.
                                                                        ; NOP may be used.
                                        ;;Send the SYNC character

000442  170451  000753  130460  TP012:  INC\L    R13,R11 @SECC   %CALL   CLK16   ;SEND SYNC PATTERN TO SERDES CLOCK 17 TIMES SO
                                                                        ; RETURN FROM CLOCK ROUTINE AT START OF WORD.

                                ;temp   bis     #bit15,cri
                                ;temp   clr     r12             %call   sdclk   ;SEND 2 WORDS. R11= 2.  @SDPF= READ CLKS

000443  034452  000412  120501          CLR     R12     @SDPF   %CALL   SDCLK   ;SEND 2 WORDS. R11= 2.  @SDPF= READ CLKS
                                                                        ; CLEAR R12 SO DATA= 1,3,7, .. ,FFFF, 0, 1, ...
000444  033757  000105  000000          MOV     #69.,R17

                                        ;;Send 255. WORDS of data (16. BITS at a time)

000445  033751  000377  120501          MOV     #255.,R11       %CALL   SDCLK   ;WORDS 3-257. LET EDC= C000H

                                        ;;Unload word 256. and set ECC time enable (@SECCT)

000446  034452  000772  010000          CLR     R12     @SECCT
000447  033753  000010  130464          MOV     #8.,R13         %CALL   CLK     ;SEND FILL BITS

                                        ;;Send WORD 257. - 265. and ECC symbols 1 - 14.

000450  033755  007061  000000          MOV     #61,DBAR,BAR                    ;ECC1 AT DBAR= 61Q.
                                                                        ; HERE IN WORD 257 BEFORE UNLOAD

000451  033751  000016  120500  TP013:  MOV     #14.,R11        %CALL   RDCLK   ;UNLOAD WORDS 257- 265.
```

```
                                                                        ; R11=14 10 BIT WORDS TO @RECCT

                                        ;;Send WORD 266. and reset ECC time enable (@RECCT)

000452  013440  000560  000000          NOP             @RECCT                  ;HERE IN WORD 266 BEFORE UNLOADED
000453  033751  000007  120500          MOV     #7.,R11         %CALL   RDCLK   ;SEND EXTRA CLOCKS TO SEE IF RSGEN CLOCKS STOP
000454  013700  000007  127777          TST     (ECC)           %RET            ;TEST FOR ECC ERROR ON RETURN. ERROR IF NEG: 1
```



```
                                ;;*****************************************************************************
                                ;
                                ;LEVO2 - LEVO3 BY RESET         CLOCKING SUBROUTINES
                                ;       CLK:
                                ;       CLK1:   CLK10:  CLK16:
                                ;
                                ;       Input:
                                ;               R13 = NUMBER OF SPECIFIED PULSES FOR THE CONTROL REGISTER
                                ;       Output:
                                ;               R13 = 0
                                ;               R12 SHIFTED RIGHT R13 TIMES.
                                ;
                                ;       REGISTERS USED:
                                ;               R11 = TESTED ON RETURN. USED FOR LOOP COUNTER
                                ;               R12 = HOLDS SERIAL DATA
                                ;               R13 = HOLDS # OF CLOCKS
                                ;               R16 = HOLDS ECC SUM
                                ;               R17 = SERDES PARALLEL DATA. (READ)= EDC. (WRITE)= WRITE DATA
                                ;
                                ;       CALLS   SD.RS, CLK, SDCLK, GENSYN, , D.GECC
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105154 - Read mode, SERDES 16, 10 RSGEN and ECC circuitry error
                                ;
                                ;;*****************************************************************************
000455  033753  000100  100464  CLK64:  MOV     #64.,R13        %JMP    CLK     ;GENERATE 64. CLOCKS
000456  033753  000040  100464  CLK32:  MOV     #32.,R13        %JMP    CLK     ;GENERATE 32. CLOCKS
000457  033753  000006  110464  CLK6:   MOV     #6.,R13         %JMP    CLK     ;GENERATE  6. CLOCKS

000460  033753  000020  100464  CLK16:  MOV     #16.,R13        %JMP    CLK     ;GENERATE 16. CLOCKS
000461  033753  000012  110464  CLK10:  MOV     #10.,R13        %JMP    CLK     ;GENERATE 10. CLOCKS
000462  033753  000001  100464  CLK1:   MOV     #1,R13          %JMP    CLK     ;GENERATE  1. CLOCK

                                        ;;Clock routine - Write = @RDPF, Read = @SDPF

000463  013453  100013  100036  CLKP:   TST     R13             %JPRDY  ERRB2H  ;ERROR IF PRDY NOT RESET BY SD16 LOAD/UNLOAD
000464  053452  010012  140505  CLK:    ROT\RF  R12             %JZRO   CLKO1   ;SHIFT DATA PATTERN
000465  133564  044010  040466          BIS\T   #DDD,CRI,DCRD   %TLSB           ;IF DATA LSB = 1, DDD TO CR
000466  133544  164004  150036          BIS\F   #DDC,CRI,DCRD   %JOVER  ERRB2H  ;ERROR IF SD16 NOT UNLOADED IN TIME
000467  135544  004004  000000          BIC     #DDC,CRI,DCRD
000470  135544  004010  000000          BIC     #DDD,CRI,DCRD                   ;CLEAR DIAGNOSTIC CLOCK/DATA
000471  031453  100013  010464          DEC     R13             %JNPRDY CLK     ;LOOP FOR NEXT CLOCK
000472  013457  170017  050464  CLKOO:  TST\F   R17             %JNWRC  CLK     ;**NOP\F

                                        ;;Load or Unload SERDES 16 when PRDY & WRC only

                                ;temp   tst     cri
                                ;temp   xor\lt  (sd),r17        %jneg   clkp    ;READ MODE - GET DATA AND CALCULATE EDC

000473  076537  060006  140463          XOR\LT  (SD),R17        %JDPF   CLKP    ;READ MODE - GET DATA AND CALCULATE EDC

                                        ;;Can save 1 WORD if can figure how to use INC\L for read mode
```

```
000474  150457  000017  000000          INC\R   R17                             ;WRITE MODE - UPDATE DATA AND LOAD SERDES
000475  013457  006017  100463          MOV     R17,\N,SD       %JMP    CLKP    ;DATA= 8000, C000, E000, ... FFFF, 0, 8000, ...
```



```
                                ;;*****************************************************************************
                                ;
                                ;       RDCLK:
                                ;       SDCLK:
                                ;
                                ;       Input:
                                ;               DBAR= BUFFER POINTER TO START OF DATA STORAGE
                                ;
                                ;       REGISTERS USED:
                                ;               R11 = USED FOR LOOP COUNTER
                                ;               R12 = HOLDS SERIAL DATA
                                ;               R13 = HOLDS # OF CLOCKS
                                ;               R15  =(DBAR) FOR BAR INCREMENT
                                ;               R16 = HOLDS ECC SUM
                                ;               R17 = SERDES PARALLEL DATA. (READ)= EDC. (WRITE)= WRITE DATA
                                ;       CALLS   CLK10
                                ;
                                ;;*****************************************************************************

                                        ;;Load RESIDUES

000476  031451  000011  120461  RDCLKO: DEC     R11             %CALL   CLK10   ;READ  MODE - SEND DATA 16 BITS AT A TIME
000477  130455  017015  127777          INC     DBAR,,BAR       %RZRO           ;IS THE LOOP DONE?
000500  033712  000003  110476  RDCLK:  MOV     (BUF),R12       %JMP    RDCLKO  ;GET NEXT ECC RESIDUE FOR SERIALIZING

                                        ;;Clock loop

                                ;SDCLK: tst     cri
                                ;       inc\r   r12             %cnneg  clk10   ;WRITE MODE - SEND DATA 10 BITS AT A TIME
                                ;       tst     cri
                                ;       dec     r11             %cneg   clk16   ;READ  MODE - SEND DATA 16 BITS AT A TIME

000501  150452  060012  030461  SDCLK:  INC\R   R12             %CNDPF  CLK10   ;WRITE MODE - SEND DATA 10 BITS AT A TIME
000502  031451  060011  130460          DEC     R11             %CDPF   CLK16   ;READ  MODE - SEND DATA 16 BITS AT A TIME

000503  120455  017015  000501  TP017:  INC     DBAR\0,,BAR     %JNZRO  SDCLK   ;IS THE LOOP DONE?

                                        ;;Return for SDCLK & CLK routines

000504  013451  000011  127777  CLK02:  TST     R11             %RET            ;R11= WORD COUNT
                                                                                ; RETURN TO SDCLK OR CLK ROUTINE
000505  030516  003007  110504  CLK01:  ADD     (ECC),R16,BUF   %JMP    CLK02   ;R16= ECC SUM FOR DATA FIELD - CLOCK AT 10 BIT RATE
                                                                                ; SAVE ECC VALUES IN BUFFER
```

```
                                ;;*****************************************************************************
                                ;
                                ;       OCLK:   ONLY CLOCK
                                ;
                                ;       Input:
                                ;               R13 = NUMBER OF SPECIFIED PULSES FOR THE CONTROL REGISTER
                                ;       Output:
                                ;               R13 = 0
                                ;               R12 SHIFTED RIGHT R13 TIMES.
                                ;
                                ;       REGISTERS USED:
                                ;               R13 = HOLDS # OF CLOCKS
                                ;
                                ;;*****************************************************************************

000506  133753  000377  100511  OCLKRS: MOV     #177400,R13     %JMP    OCLK    ;GENERATE LOTS OF CLOCKS FOR RESET
000507  033753  000060  110511  OCLK48: MOV     #48.,R13        %JMP    OCLK    ;GENERATE 48. CLOCKS
000510  033753  000001  100511  OCLK1:  MOV     #1.,R13         %JMP    OCLK    ;GENERATE  1. CLOCK

000511  133544  014004  167777  OCLK:   BIS\F   #DDC,CRI,DCRD   %RZRO            ;SET DIAGNOSTIC CLOCK
000512  135544  004004  000000          BIC     #DDC,CRI,DCRD                   ;RESET DIAGNOSTIC CLOCK
000513  031453  000013  110511  OCLK00: DEC     R13             %JMP    OCLK    ;NEXT CLOCK PLEASE /
                                                                                ; DONE WITH ALL CLOCKS?

000411                          START   = TP010
000504                          ONLY    = CLK02         ;WRITE MODE

                                ;       SYM2 =  H       SYM3 =  H       SYM4 =  H       SYM5 =  H
                                ;       SYM6 =  H       SYM7 =  H       SYM8 =  H       SYM9 =  H
                                ;       SYM10 = H       SYM11 = H       SYM12 = H       SYM13 = H
                                ;       SYM407= H       SYM408= H       SYM409= H       SYM410= H
                                ;       SUM =   H

000417                          START   = TP011
000504                          ONLY    = CLK02         ;WRITE MODE

                                ;       ACTUAL ECC SYMBOLS ARE THE LOWER 10 BITS
                                ;       ECC-2 = ED89H   ECC-1 = EF62H
                                ;
                                ;       ECC01 = 8F29    ECC02 = 8F5A    ECC03 = 8CC7    ECC04 = 8DB3
                                ;       ECC05 = 8C01    ECC06 = 8D37    ECC07 = 8CB1    ECC08 = 8C79
                                ;       ECC09 = 8C24    ECC10 = 8C68    ECC11 = 8F8E    ECC12 = 8DA2
                                ;       ECC13 = 8E9F ?? ECC14 = 8C81    GARBAGE= 000F   ECC15 = 8F15
                                ;       ECC16 = 8CD3    ECC17 = 8F62

000417                          START   = TP011         ;TO TRIGGER ANALYSER AT START OF ECC SYMBOL 1
000503                          THEN    = TP017         ;DATA= 000DH
000472                          THEN    = CLK00         ;DATA= 0006H
```



```
000430                          START   = SDRD
000451                          THEN    = TP013
000476                          ONLY    = RDCLK0        ;READ  MODE LOADING SERIAL DATA

                                ;       ACTUAL ECC SYMBOLS ARE THE LOWER 10 BITS
                                ;
                                ;       ECC01 = 8F29    ECC02 = 8F5A    ECC03 = 8CC7    ECC04 = 8DB3
                                ;       ECC05 = 8C01    ECC06 = 8D37    ECC07 = 8CB1    ECC08 = 8C79
                                ;       ECC09 = 8C24    ECC10 = 8C68    ECC11 = 8F8E    ECC12 = 8DA2
                                ;       ECC13 = 8E9F ?? ECC14 = 8C81    ECC15 = 8F15    ECC16 = 8CD3
                                ;       ECC17 = 8F62

000442                          START   = TP012
000504                          ONLY    = CLK02         ;READ MODE

                                ;       READ  MODE
                                ;
                                ;       SYM2 =  H       SYM3 =  H       SYM4 =  H       SYM5 =  H
                                ;       SYM6 =  H       SYM7 =  H       SYM8 =  H       SYM9 =  H
                                ;       SYM10 = H       SYM11 = H       SYM12 = H       SYM13 = H
                                ;       SYM407= H       SYM408= H       SYM409= H       SYM410= H
                                ;       SUM =   H
```

```
                                ;;*****************************************************************************
                                ;
                                ;       SD.RST:         SERDES RESET ROUTINE
                                ;
                                ;
                                ;       REGISTERS USED:
                                ;               R13 = HOLDS # OF CLOCKS
                                ;               R17 = SERDES PARALLEL DATA. (READ)= EDC. (WRITE)= WRITE DATA
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105153 - Write mode, wrap SERDES 16 error
                                ;       105154 - Read mode, SERDES 16, 10 RSGEN and ECC circuitry error
                                ;
                                ;;*****************************************************************************

                                        ;;Initialize SERDES and DIAGNOSTIC HARDWARE (ERROR CODE = 8.)

000514  013440  000520  120250  SD.RST: NOP             @RRM    %CALL   INCRTN  ;SET TO WRITE MODE
000515  034457  000577  120506          CLR     R17     @RECCT  %CALL   OCLKRS  ;RESET ECC TIMING
000516  013440  000500  000000  SD.RS:  NOP             @RSE                    ;
000517  013440  100540  110036          NOP             @RECC   %JPRDY  ERRB2H  ;ERROR IF PRDY NOT RESET BY @RSE
000520  013440  160620  110036          NOP             @SRSGEN %JOVER  ERRB2H  ;ERROR IF LATE NOT RESET BY @RSE
                                                                                ; CRY MUST BE A 1 TO SEE WRC
000521  013440  000420  000000          NOP             @RRSGEN
000522  013440  120460  100036  SDRRDY: NOP             @SCLR   %JRRDY  ERRB2H  ;ERROR IF RRDY NOT RESET BY @RECC & @RRSGEN
000523  013440  140660  110036          NOP             @RCLR   %JDSER  ERRB2H  ;ERROR IF DSER NOT RESET BY @SCLR & CLOCKS
                                                                                ; RESET I/O CLEAR.

                                ;;*****************************************************************************
                                ;
                                ;       TSTRTN ROUTINE
                                ;
                                ;;*****************************************************************************

000524  013451  000671  127777  TSTRTN: TST     DER     @RCLR   %RET            ;RESET I/O CLEAR / RETURN
                                                                                ; THIS RETURN IS USED BY OTHERS
                                                                                ; FOR THE SAME PURPOSE: SET RETURN STATUS
                                                                                ; TO SEE IF THE NEXT TEST IS EXECUTED
```



```
                                ;;*****************************************************************************
                                ;
                                ;       TEST %RRDY SUBROUTINE
                                ;       @SECC, @SECCT => ECC START H = 1
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       105153 - Write mode, wrap SERDES 16 error
                                ;       105154 - Read mode, SERDES 16, 10 RSGEN and ECC circuitry error
                                ;
                                ;;*****************************************************************************

000525  133544  124004  110036  D.RRDY: BIS     #DDC,CRI,DCRD   %JRRDY  ERRB2H  ;ECC SYMBOL CLK L POSITIVE TRANSITION
000526  013472  000012  040000          TST\T   R12
000527  135544  004004  000000          BIC     #DDC,CRI,DCRD
000530  053452  000012  100522          ROT\R   R12             %JMP    SDRRDY  ;TEST IF RRDY RESET BY INSTRUCTION EXECUTION

000531  013440  000720  000000  S.SRM:  NOP             @SRM                    ;SET READ MODE
000532  016513  000546  127777          XOR     (SD),R13\N @RECC %RET           ;TEST WRAP DATA AND RETURN

                                ;;*****************************************************************************
                                ;
                                ;LEVO2
                                ;       GENERATE THE SYNC PATTERN
                                ;
                                ;       Output:
                                ;               R12 = SYNC PATTERN 023274 OR 26BCH
                                ;
                                ;;*****************************************************************************

000533  033752  000274  000000  GENSYN: MOV     #SYNCL,R12              ;
000534  133552  000046  127777          BIS     #SYNCH,R12      %RET    ;R12 = SYNC = 023274= 26BCH= 9916.
```

```
                                ;;*****************************************************************************
                                ;
                                ;LEVO2
                                ;       RESRED
                                ;               SEND ECC SYMBOLS (STORED IN RAM) THROUGH SERDES 10
                                ;               FOR READ ECC TEST AND READ RESIDUES
                                ;
                                ;       Input:
                                ;               R11 = LOOP COUNT
                                ;               DBAR = WHERE YOU WANT TO GET ECC SYMS FROM THE BUFFER
                                ;       Output:
                                ;               R12 WILL BE ALTERED
                                ;               DBAR GETS INCREMENT BY # TIMES THROUGH LOOP
                                ;               BUF GETS NEW ECC'S
                                ;
                                ;       REGISTERS USED:
                                ;               DBAR = ADDRESS OF RAM BUFFER
                                ;               R11 = USED FOR LOOP COUNTER
                                ;               R12 = HOLDS SERIAL DATA
                                ;               R13 = HOLDS # OF CLOCKS
                                ;               R14 = TEMP & HOLDS ECC REGISTER VALUE
                                ;               R15  =(DBAR) FOR BAR INCREMENT
                                ;               R16 = HOLDS ECC SUM
                                ;               R17 = SERDES PARALLEL DATA. (READ)= EDC. (WRITE)= WRITE DATA
                                ;
                                ;       CALLS   SD.RS, CLK, SDCLK, GENSYN, GETECC, D.GECC, CLOK10
                                ;
                                ;;*****************************************************************************

000432                          START   = TP014         ;RESIDUES = 0
000536                          ONLY    = TP015         ;RES01 - RES17 = 0

                                ;

000432                          START   = TP016         ;RESIDUES = ERROR RESIDUES
000536                          ONLY    = TP015

                                ;       FOLLOWING RESIDUES SHOULD BE SAVED IN REGEN:
                                ;
                                ;       RES01 =         RES02 =         RES03 =         RES04 =
                                ;       RES05 =         RES06 =         RES07 =         RES08 =
                                ;       RES09 =         RES10 =         RES11 =         RES12 =
                                ;       RES13 =         RES14 =         RES15 =         RES16 =
                                ;       RES17 =
```



```
                                ;;*****************************************************************************
                                ;
                                ;LEVO2
                                ;       GET ECC
                                ;
                                ;;*****************************************************************************

000535  120455  007015  133002  GETECO: INC     DBAR\0,,BAR     %CALL   D.GECC  ;UPDATE BAR AND GET ECC IN R14
                                                                                ; WITH UPPER BITS CLEARED
000536  020156  003014  130461  TP015:  ADD     R14\0,R16,BUF   %CALL   CLK10   ;SAVE RESIDUES

000537  013476  010016  177777  GETECC: TST\T   R16             %RZRO           ;TEST R16= ECC SUM FOR RETURN
000540  031451  000011  110535          DEC     R11             %JMP    GETECO  ;

000541                                  .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO SEQERR
                                        .ORG    700
                                        .PAGE
```

```
                                        .SBTTL  D.PROC IDLE LOOP AND OP CODE PROCESSOR(*** NEW SDI TIMING ***)
                                    ;;+     09-JULY-83      UDA52 MICROCODE
                                    ; ROUTINE NAME:
                                    ;       D.IDLE (DRIVE IDLE LOOP)
                                    ;
                                    ;  FUNCTIONAL DESCRIPTION:
                                    ;       THIS CODE IS THE MAIN SCHEDULING LOOP FOR THE DRIVE PROCESSOR AND
                                    ; AS SUCH IT PERFORMS THE FOLLOWING FUNCTIONS:
                                    ;       1. MONITOR SDI INTERCONNECT DRIVE STATE
                                    ;       2. MONITORS 'DMODE' FLAG OF SYSTEM STATUS AND SHORTENS IDLE LOOP
                                    ;          ACCORDINGLY IF IT IS SET
                                    ;       3. MONITORS SDI INTERCONNECT STATUS WORDS FOR DRIVE PROCESSOR ACTIONS
                                    ;          REQUIRED
                                    ;
                                    ; INPUTS:
                                    ;       1. SDI.ST               SDI CONTROL BLOCK STATUS
                                    ;
                                    ; OUTPUTS:
                                    ;       1. SDI.ST               SDI CONTROL BLOCK STATUS
                                    ;
                                    ; EQUATES USED:
                                    ;       *** FOR SDI STATUS ***
                                    ;       PKIP    := 000001       ; 0 = NO PKT, 1 = PACKET IN PROGRESS
                                    ;       SEEK    := 000002       ; 0 = NO SEEK NEEDED, 1 = SEEK INITIATE NEEDED
                                    ;       DERR    := 000004       ; 0 = NO ERROR, 1 = D.PROC ERROR ON I/O COMMAND
                                    ;       VECT    := 000010       ; 0 = VECTOR LEVEL DONE, 1 = STATE VECTOR LEVEL ACTIVE
                                    ;       BFRQ    := 000020       ; 0 = OK, 1 = BUFFER SPACE/PARAMETERS REQUESTED
                                    ;       SUSP    := 000040       ; 0 = TASK RUNNING, 1 = TRANSFER TASK SUSPENDED
                                    ;       BFSV    := 000100       ; 0 = OK, 1 = BUFFER SERVICE REQUIRED
                                    ;       XCMP    := 000200       ; 0 = OK, 1 = TRANSFER TASK COMPLETED
                                    ;       DATT    := 000400       ; 0 = OK, 1 = DRIVE ATTENTION PENDING
                                    ;               := 001000       ; NOT USED
                                    ;       DRDUP   := 002000       ; DRDUP, DRVOL, AND DRAVL ARE ENCODED AS FOLLOWS:
                                    ;                                 DDD
                                    ;                                 RRR
                                    ;                                 AVD
                                    ;                                 VOU
                                    ;                                 LLP
                                    ;       DRVOL   := 004000       ; 000 = DRIVE ONLINE
                                    ;       DRAVL   := 010000       ; 001 = DRIVE ONLINE AND DUPLICATE
                                    ;                               ; 010 = DRIVE OFFLINE AND INOPERATIVE
                                    ;                               ; 011 = DRIVE OFFLINE AND DUPLICATE
                                    ;                               ; 100 = DRIVE AVAILABLE (SPINABLE OR NOT SPINABLE)
                                    ;                               ; 101 = NOT USED
                                    ;                               ; 110 = NOT USED
                                    ;                               ; 111 = DRIVE OFFLINE AND UNKNOWN TO CONTROLLER
                                    ;       SLAT    := 020000       ; 0 = NOP, 1 = SEND LOG OR ATTENTION PKT TO HOST
                                    ;                               ; IN ADDITION SLAT IS USED DURING INTIALIZATION
                                    ;                               ; TO INIT EACH DRIVE
                                    ;       ERRIP   := 040000       ; 0 = NO ERROR RECOVERY, 1 = LEVEL 0 RECOVERY ACTIVE
                                    ;                               ; IN ADDITION ERRIP IS USED AS A FLAG TO AID IN
                                    ;                               ; DETERMINING IF A DRIVE IS ALREADY ONLINE TO THE UDA
                                    ;       RVCT    := 100000       ; 0 = NO REVECTORING REQUIRED, 1 = REVECTORING REQUIRED
                                    ;       *** FOR REAL TIME DRIVE STATE ***
```



```
                                    ;       DRDY    := 000001       ; 0 = DO NOT SEND, 1 = DRIVE RECEIVER READY FOR COMMAND
                                    ;       ATTN    := 000002       ; 0 = NO ERROR,  1 = ATTENTION CONDITION FOUND
                                    ;       SENT    := 000004       ; 0 = FRAME NOT SENT, 1 = RTCS FRAME SENT
                                    ;       RCVD    := 000010       ; 0 = DRIVE STATE NOT RECVD, 1 = DRIVE STATE RECVD
                                    ;       SEC     := 000020       ; 0 = NO SECTOR, 1 = SECTOR PULSE(USE 1-0 EDGE)
                                    ;       IDX     := 000040       ; 0 = NO INDEX, 1 = INDEX PULSE (USE 1-0 EDGE)
                                    ;       AVAIL   := 000100       ; 0 = NOT AVAILABLE, 1= DEVICE AVAILABLE FOR USE
                                    ;       RRDE    := 000200       ; 0 = NO ERROR, 1 = DATA TRANSMIT ERROR
                                    ;       DPE     := 000400       ; 0 = NO ERROR, 1 = DATA PARITY ERROR
                                    ;       RWRDY   := 100000       ; 0 = NOT R/W READY, 1 = READ/WRITE READY
                                    ;
                                    ;;-
000700  135544  010020  013626  D.IDLA: BIC     #DXFC,RLL       %JNZRO  DMNTRY  ; IF DMBEG EQ 1 THEN START DM   [rae3-c119]
                                                                            ; and clear lock for Uproc      [rae3-c119]
000701  114544  070100  000701  D.IDLE: BIT     #DMODE,RLL      %JACLO  D.IDLE  ; [16K]IF LO POWER THEN WAIT/IF IN DM MODE
000702  114544  010040  010700          BIT     #DMBEG,RLL      %JNZRO  D.IDLA  ; [16K]  THEN CHECK FOR DM START
                                    ; ***   SELECT SDI INTERCONNECT AND SDI CONTROL BLOCK POINTER ***
000703  073717  000001  123600          MOV\L   (DCRS),R17      %CALL   S.SD11  ; [16K]R17=CONTROL REG IMAGE/R11=PTR TO 1ST SDI BLK

000704  034557  000016  000000  D.RAMX: AND     #SDIS-1,R17                     ; [16K][U52EC1]ISOLATE SDI SELECT
000705  033777  010001  150706          MOV\T   #BIT00,R17      %TZRO           ; IF EQ 0 THEN RESET TO 1ST SDI
000706  033455  000011  121007          MOV     R11,DBAR        %CALL   X.SSDX  ; DBAR=PTR TO 1ST SDI CNTL BLK
000707  053457  000017  010000  D.IDLB: SHF\R   R17                             ; ROTATE R17 RIGHT
000710  030575  040120  150707          ADD\T   #SDIB.L,DBAR    %JNLSB  D.IDLB  ; IF MSB NEQ 1 THEN UPDATE DBAR TO NXT CNTL BLK
                                    ; ***   CHECK CURRENT SDI STATUS OF DRIVE ***
000711  010555  007000  121006          ADD     #SDI.ST,DBAR\N,BAR %CALL X.SSDT ; SEND RTCS FRAME
000712  033712  000002  010000          MOV     (RTDS),R12                      ; R12=DRIVE STATUS
000713  033711  000003  000000          MOV     (BUF),R11                       ; R11=SDI STATUS
000714                                  ASSUME  PKIP,EQ,BIT00
000714  114551  040030  000752          BIT     #<DRVOL+DRAVL>,R11 %JLSB D.IDLM ; IF PKT ACTIVE THEN GO CHK STATUS
                                    ; ***   DRIVE OFFLINE/AVAILABLE/UNKNOWN - CHECK REAL TIME DRIVE STATUS ***
000715  027456  010011  110747          COM     R11\0,R16       %JZRO   D.IDLJ  ; IF DRIVE ONLINE THEN GO CHECK DRIVE STATE
000716  114556  140034  100701          BIT     #<DRVOL+DRAVL+DRDUP>,R16 %JDSER D.IDLE ; IF PARITY ERROR THEN GO IDLE
000717  114551  010010  010732          BIT     #DRVOL,R11      %JNZRO  D.IDLD  ; IF NOT UNKNOWN THEN GO CHK FOR OFFLINE/AVAIL
000720  114556  110040  010701          BIT     #SLAT,R16       %JNCSR  D.IDLE  ; IF NO STATE THEN GO IDLE
                                    ; ***   DRIVE 'UNKNOWN' WITH REAL TIME DRIVE STATE ***
                                    ; ***   CHECK FOR POWER UP DRIVE INIT NEEDED ***
000721  135571  013040  161022          BIC\T   #SLAT,R11,BUF   %CZRO   D.INIT  ; IF INIT NEEDED THEN SEND 1
000722  034552  000100  000000          AND     #AVAIL,R12                      ; IF AVAIL SET
000723  134551  010100  040742          AND\F   #ERRIP,R11      %JNZRO  D.IDLG  ;   THEN DO GET STATUS
000724  033452  010531  150701          MOV\F   R11,R12 @SWM    %JZRO   D.IDLE  ; IF ERRIP NOT SET THEN GO IDLE
000725  034457  000457  123572          CLR     R17     @SCMD   %CALL   D.GMST  ; SET COMMAND MODE/R11=MUTEXED SDI STATUS
000726  035151  003712  133570          BIC     R12,R11,BUF @SSE %CALL  UNLOCK  ; ZAP ERRIP/RELEASE U.PROC
000727  013460  010500  140701  D.IDLC: NOP\T            @RSE    %JZRO   D.IDLE  ; IF DONE THEN GO IDLE
000730  031457  100017  000727          DEC     R17             %JNDCLK D.IDLC  ; IF NO DRV CLK DECREMENT R17 AND LOOP
000731  034457  006517  100742          CLR     R17,R17,SD @RSE %JMP    D.IDLG  ; ZAP SERDES/RESET SERDES/DO GET STATUS
                                    ; ***   DRIVE 'OFFLINE' (INOPERABLE/DUPLICATE) - CHECK FOR CHANGE IN STATE ***
000732  034552  010102  100736  D.IDLD: AND     #<ATTN+AVAIL>,R12 %JZRO D.IDLF  ; IF AVAILABLE THEN CONTINUE
000733  014552  000100  010000          BIT     #AVAIL,R12                      ; [U52EC2]IF RTDS AVAIL LO
000734  013440  010000  100745          NOP                     %JZRO   D.IDLI  ; [U52EC2]  THEN MARK OFFLINE/UNKNOWN
000735  013440  110000  110701          NOP                     %JCSR   D.IDLE  ; IF STATE THEN GO IDLE
                                    ; ***   DRIVE 'AVAILABLE' (SPINABLE OR NOT SPINABLE) ***
000736  037714  110002  000745  D.IDLF: COM     (RTDS),R14      %JNCSR  D.IDLI  ; IF NO STATE THEN MARK OFFLINE/ELSE R14=-DRIVE STATE
```

```
000737  010555  007044  133603          ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD11 ; R11=DRIVE STATUS
000740                                  ASSUME  DRV.RU,EQ,BIT00                 ; MAKE SURE DRV.RU IS LSB
000740  036552  040102  040743          XOR\F   #<ATTN!AVAIL>,R12 %JLSB D.IDLH  ; IF SPINABLE THEN CONTINUE
000741  013440  010000  010701          NOP                     %JNZRO  D.IDLE  ;   ELSE NOT SPINABLE/IF STILL NOT SPINABLE GO IDLE
                        ; ***   FORCE UDA TO DO GET STATUS ***
000742  033752  000002  110752  D.IDLG: MOV     #ATTN,R12               %JMP    D.IDLM  ;   ELSE FORCE GET STATUS AND DISCONNECT
000743  034554  000102  010000  D.IDLH: AND     #<ATTN!AVAIL>,R14               ; IF STILL SPINABLE
000744  013440  010000  100701          NOP                     %JZRO   D.IDLE  ;   THEN GO IDLE
                        ; ***   MARK DRIVE OFFLINE ***
000745  013440  000000  122045  D.IDLI: NOP                     %CALL   D.OFFD  ; GO MARK OFFLINE/UNKNOWN
000746  013440  000000  110701          NOP                     %JMP    D.IDLE  ; GO IDLE
                        ; ***   'MSCP ONLINE' CHECK IF DRIVE SANE ***
000747  033712  110002  010745  D.IDLJ: MOV     (RTDS),R12      %JNCSR  D.IDLI  ; [U52EC2]IF NOT ALIVE THEN MARK OFFLINE
000750  014552  140100  100701          BIT     #AVAIL,R12      %JDSER  D.IDLE  ; [U52EC2]IF PARITY ERROR THEN GO IDLE
000751  013440  010000  010745          NOP                     %JNZRO  D.IDLI  ; IF AVAIL THEN MARK OFFLINE
                        ; ***   CHECK FOR DRIVE ATTENTION ***
000752  014552  000100  010000  D.IDLM: BIT     #AVAIL,R12                      ; IF AVAILABLE SET
000753  014552  010002  010755          BIT     #ATTN,R12       %JNZRO  D.IDLN  ;   THEN CONTINUE/IF DRIVE ATTENTION UP
000754  133773  010001  062013          MOV\T   #DATT,R13       %CNZRO  D.SSET  ;   THEN SET SDI STATUS BIT
                        ; ***   CHECK FOR WORK TO DO FOR THIS SDI CONTROL BLOCK ***
000755  010555  007022  123610  D.IDLN: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD16 ; R16=MSCP PKT PTR FOR D.PCMD & D.SEEK
000756  014551  000001  010000          BIT     #PKIP,R11                       ; IF NOT, CHECK FOR PKT IN PROGRESS
000757  114551  010040  100701          BIT     #SLAT,R11       %JZRO   D.IDLE  ; IF NO PKT ACTIVE THEN IDLE
000760  114551  010300  000701          BIT     #<RVCT!ERRIP>,R11 %JNZRO D.IDLE ; [16K]IF LOG/ATTN FLAG SET THEN GO IDLE
000761  014571  010272  150762          BIT\T   #<SUSP!VECT!XCMP!BFRQ!SEEK>,R11 %TZRO ; [16K]IF REVECT/ERROR RECOV THEN FORCE SUSPEND
000762  014551  010010  111034          BIT     #VECT,R11       %JZRO   D.PCMD  ; IF NOT SUSPENDED THEN GO PROCESS CMD
000763  014551  010002  011170          BIT     #SEEK,R11       %JNZRO  D.VECT  ; IF VECTOR STATE IN PROGRESS THEN GO DO IT
000764  114551  010200  011163          BIT     #RVCT,R11       %JNZRO  D.SEEK  ; IF SEEK REQ'D THEN GO DO IT
000765  114551  010100  011142          BIT     #ERRIP,R11      %JNZRO  D.RVCT  ; IF REVECTORING THEN GO DO IT
000766  013440  010000  100701          NOP                     %JZRO   D.IDLE  ; IF NOT ERROR RECOVERY THEN GO IDLE
000767                                  ASSUME  D.ERR1,EQ,.                     ; MAKE SURE ERROR RECOVERY IS NEXT
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.ERR1 (D.PROC LEVEL 1 ERROR ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SERVE AS THE BASE PROCESS WHICH WILL VECTOR
                                ; INTO THE VECTOR TABLE AT OFFSET 'V.ERR1' IN ORDER TO PERFORM THE
                                ; FUNCTIONS NECESSARY TO PERFORM LEVEL 1 ERROR RECOVERY.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       LEVEL 1 ERROR RECOVERY STEP PERFORMED
                                ;
                                ;;-

000767  010555  007017  133605  D.ERR1: ADD     #SDI.E1,DBAR\N,BAR %CALL S.LD13 ; R13=LEVEL 1 ERROR STATE
                        ; ***   IF FIRST RETRY THEN SEND ERROR RECOVERY COMMAND ***
000770  013440  010000  100776          NOP                     %JZRO   D.ER1A  ; [EERREC]
000771  010555  007006  133603          ADD     #SDI.UB,DBAR\N,BAR %CALL S.LD11 ; [EERREC] GET BUFFER DESCRIPTOR POINTER
000772  030551  007001  123603          ADD     #BUF.ST,R11,BAR %CALL   S.LD11  ; [EERREC] GET BUFFER STATUS WORD
000773  114551  000001  000000          BIT     #BERDN,R11                      ; [EERREC] CHECK IF ALREADY ISSUED FOR THIS LEVEL
000774  134553  010016  010776          AND     #<RETCNT&<RETCNT-1>>,R13 %JNZRO D.ER1A ;[EERREC][ECO#1]THE DON'T ISS CMD/IF RETRY COUNT <= 1
000775  033751  010305  111166          MOV     #V.ERR1&LOBYT,R11 %JZRO D.VECP  ;   THEN R11=LO VECTOR TABLE ADDR/GO INIT VECTOR
                        ; ***   ERROR RECOVERY COMMAND COMPLETE ***
000776  033753  000050  132011  D.ER1A: MOV     #<VECT+SUSP>,R13 %CALL  D.CLRS  ; R13=FLAGS TO CLEAR/GO CLEAR THEM
                        ; ***   RETRY THE READ OPERATION ***
000777  010555  007006  133611          ADD     #SDI.UB,DBAR\N,BAR %CALL S.LD17 ; R17=PTR TO BUFFER IN ERROR
001000  010555  007026  123603          ADD     #SDI.ES,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=EXTENDED STATUS
001001  033551  003040  010000          BIS     #URETRY,R11,BUF                 ; [U52EC2]INDICATE U.PROC RETRY
                        ; ***   ENTER HERE TO PERFORM READ OF BUFFER POINTED TO BY R17 ***
001002  013440  000000  132476  D.ERRD: NOP                     %CALL   D.BFCA  ; [ECO#2]GO CHK BUFFER STATUS
001003  013451  000611  112602          TST     R11     @RDPF   %JMP    D.REDA  ; TEST SDI STATUS/GO REREAD SECTOR/GO IDLE
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       X.SSDI
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SET THE SDI INTERCONNECT SELECT IN CR
                                ; FROM THE CONTENTS OF DM REGISTER 2.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 2           SDI INTERCONNECT SELECT
                                ;
                                ; OUTPUTS:
                                ;       CR UPDATED TO NEW SDI INTERCONNECT
                                ;
                                ;;-

001004  013740  007002  123611  X.SSDI: MOV     #DMREG2,BAR     %CALL   S.LD17  ; R17=SDI INTERCONNECT SELECT
001005  034557  000017  131007          AND     #SDIS,R17       %CALL   X.SSDX  ; ISOLATE LO 4 BITS OF R17
001006  013740  005001  101015  X.SSDT: MOV     #CONT,RTCS      %JMP    X.SSTA  ; SET/GO SEND CONTROLLER STATE
                                ; ***   PREPARE CURRENT SDI PORT FOR SWITCHING ***
001007  033713  000001  131012  X.SSDX: MOV     (DCRS),R13      %CALL   X.SSDS  ; [16K]R13=CURRENT PORT SELECT BITS
001010  035553  000017  010000          BIC     #SDIS,R13                       ; [16K]CLEAR PORT SELECT BITS
001011  033153  004017  010000          OR      R17,R13,DCRD                    ; [16K]RESET CR (SELECT NEW PORT)

                                ; ***   SYNCHRONIZE WITH CURRENT SDI PORT (31 - 48 BIT TIMES) ***
                                ; ***   NOTE - THIS ROUTINE IS OPTIMIZED NOT TO WASTE TIME IN THE CASE WHERE
                                ; ***   THE DRIVE HAS NO STATUS CLOCKS (FOR EXAMPLE NO SDI CABLE) ***
001012  014440  005011  131015  X.SSDS: CLR     R11,RTCS        %CALL   X.SSTA  ; CLEAR UDA STATUS FRAME
001013  014460  015011  071015          CLR\T   R11,RTCS        %CNZRO  X.SSTA  ; CLEAR UDA STATUS FRAME IF DRIVE EXISTS
001014  014440  015011  177777          CLR\F   R11,RTCS        %RZRO           ; CLEAR UDA STATUS FRAME IF DRIVE EXISTS
                                ;       ROUTINE TO TIME OUT AN SDI STATUS FRAME FROM THE DRIVE
                                ;       RETURNED CC'S ARE NON-ZERO IF FRAME RECEIVED, ZERO IF TIMEOUT
001015  033754  000012  101020  X.SSTA: MOV     #10.,R14        %JMP    RTDSER  ; R14=SDI PORT TIMEOUT
001016  031454  110014  177777  XSSTAA: DEC\F   R14             %RCSR           ; RETURN IF SENT / DECR TIMEOUT
001017  013454  010014  167777          TST\F   R14             %RZRO           ; [ECO#1]-RETURN IF TIMEOUT WITH ZRO= 1
001020  013714  140462  041016  RTDSER: MOV\F   (RTDS),R14\N @SCLR %JNDSER XSSTAA ; IF PARITY ERR CLEAR IT AND FORCE FAIL
                                ; ***   RTDS ERROR - TEST FOR DATA/STATUS CHANNEL ERROR
001021  013454  000674  111016          TST     R14     @RCLR   %JMP    XSSTAA  ; OK, RESET & CONTINUE
                                ; ***   PASS TIMING INFORMATION FOR D.INIT ***
000003                          MINSST  := 3            ; MINIMUM CYCLES THROUGH X.SSDT (25 MHZ)
000037                          MAXSST  := 31.          ; MAXIMUM CYCLES THROUGH X.SSDT (NO CLOCKS)
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.INIT (DRIVE INITIALIZE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL ASSERT THE 'INIT' BIT OF THE RTCS FOR AT LEAST 64
                                ; STATE BIT TRANSMISSION TIMES. THE ROUTINE WAITS FOR DRIVE STATE TRANSITIONS
                                ; TO CEASE AND THEN WAITS FOR STATE TRANSITIONS TO BEGIN AGAIN. BOTH OF THE
                                ; WAITS ARE TIMED AND WILL ABORT THE INIT SEQUENCE IF TIMED OUT.
                                ;
                                ; INPUTS:
                                ;       CONTROL REGISTER (CR) SET TO THE DESIRED SDI INTERCONNECT
                                ;
                                ; OUTPUTS:
                                ;       INIT GENERATED TO THE DRIVE
                                ;       CONDITION CODES ARE NZ IF DRIVE SEEMED TO OBEY THE INIT
                                ;
                                ;;-

001022  013740  005021  121015  D.INIT: MOV     #<CONT+INI>,RTCS %CALL  X.SSTA  ; SEND INIT TO DRIVE
                                ;       MOV     #<CONT+INI>,RTCS %CALL  X.SSTA  ; [U52EC2]SEND INIT TO DRIVE
                                ; ***   WAIT 10 MSEC FOR DRIVE STATE TRANSITIONS TO STOP ***
001023  153753  000050  010000          MOV\R   #<<30303./<MINSST+3>+127.>*2&HIBYT>,R13 ; [U52EC2]R13=APPROX 10 MSEC TIMER
001024  013740  005021  121015  DINITA: MOV     #<CONT!INI>,RTCS %CALL  X.SSTA  ; [U52EC2]SET INIT, TEST FOR DRV CLKS
001025  031453  110013  011027          DEC     R13             %JNCSR  DINITB  ; [U52EC2]IF NO CLKS THEN CONTINUE/ELSE DECR R13
001026  013440  010000  001024          NOP                     %JNZRO  DINITA  ; IF NOT 10 MSECS THEN LOOP
                                ; ***   DRIVE STATE TRANSITIONS STOPPED OR TIMED OUT ***
                                ; ***   NOW WAIT 5 MSEC FOR STATE TRANSITIONS TO START AGAIN ***
001027  153753  000004  000000  DINITB: MOV\R   #<<15152./<MAXSST+3>+127.>*2&HIBYT>,R13 ; [U52EC2]R13=APPROX 5 MSEC TIMER
001030  034452  000012  131006  DINITC: CLR     R12             %CALL   X.SSDT  ; SEND CONT/WAIT FOR IT TO CLK OUT
001031  031453  110013  127777          DEC     R13             %RCSR           ; IF CLKS THEN RETURN/ELSE DECR R13
001032  013440  010000  001030          NOP                     %JNZRO  DINITC  ; IF NOT 5 MSECS THEN LOOP
001033  013440  000000  127777          NOP                     %RET            ;   ELSE TIMEOUT AND RETURN
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.PCMD (PROCESS COMMAND)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE IS ENTERED WHEN THERE IS A PACKET IN PROGRESS (BIT PKIP
                                ; SET IN THE SDI STATUS) AND THE PROCESS IS NOT SUSPENDED. THE OP CODE IS
                                ; OBTAINED FROM THE ACTIVE PACKET AND THE APPROPRIATE PROCESSING ROUTINE
                                ; IS ENTERED.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER SDI CONTROL BLOCK
                                ;       R16             POINTER TO ACTIVE PACKET
                                ;
                                ; OUTPUTS:
                                ;       VECTOR TO APPROPRIATE ROUTINE AND PROCESSED
                                ;
                                ;;-

001034  010556  007006  133612  D.PCMD: ADD     #P.OPCD,R16\N,BAR %CALL S.LL12  ; R12=OP CODE WORD
001035  010555  007025  123605          ADD     #SDI.OM,DBAR\N,BAR %CALL S.LD13 ; [U52EC1]R13=OVERLAPPED CMD
001036  013760  017156  173605          MOV\T   #NSEEKS,BAR     %CZRO   S.LD13  ; [U52EC1]R13=NUMBER OF SEEKS TO START
001037  013440  010600  010701          NOP             @RDPF   %JNZRO  D.IDLE  ; [16K]IF ANY TO START THEN GO IDLE
001040                                  ASSUME  OP.ACC,EQ,20                    ; MAKE SURE ACCESS IS 20
001040                                  ASSUME  OP.CMP,EQ,40                    ; MAKE SURE COMPARE IS 40
001040                                  ASSUME  OP.ERS,EQ,22                    ; MAKE SURE ERASE IS 22
001040                                  ASSUME  OP.RD,EQ,41                     ; MAKE SURE READ IS 41
001040                                  ASSUME  OP.RPL,EQ,24                    ; MAKE SURE REPLACE IS 24
001040                                  ASSUME  OP.WR,EQ,42                     ; MAKE SURE WRITE IS 42
001040  014552  000060  000000          BIT     #60,R12                         ; IF NOT I/O COMMAND
001041  014552  010002  111045          BIT     #2,R12          %JZRO   D.PCMB  ;  THEN GO PROCESS/IF WRITE FAMILY
001042  014552  010004  012123          BIT     #4,R12          %JNZRO  D.WRIT  ;  THEN GO PROCESS/IF READ FAMILY
001043  013440  010000  112600          NOP                     %JZRO   D.READ  ;  THEN GO PROCESS/ELSE MUST BE REPLACE
001044  013440  000000  111061          NOP                     %JMP    D.RPLC  ; AND GO DO REPLACE
001045  016552  000011  010000  D.PCMB: XOR     #OP.ONL,R12\N                   ; IF ONLINE
001046  016552  010010  111055          XOR     #OP.AVL,R12\N   %JZRO   D.ONLN  ;  THEN GO PROCESS/IF AVAILABLE
001047  016552  010013  101054          XOR     #OP.DAP,R12\N   %JZRO   D.AVAL  ;  THEN GO PROCESS/IF ACCESS PATHS
001050  016552  010012  111165          XOR     #OP.SUC,R12\N   %JZRO   D.TOPO  ;  THEN GO PROCESS/IF SET UNIT CHAR
001051  016552  010016  111164          XOR     #OP.GST,R12\N   %JZRO   D.SUCH  ;  THEN GO PROCESS/IF GET STATUS
001052  033771  010327  151166          MOV\T   #V.GSTA&LOBYT,R11 %JZRO D.VECP  ;  THEN GO PROCESS/ELSE MUST BE ATTENTION
001053                                  ASSUME  D.ATTN,EQ,.                     ;  AND FALL INTO D.ATTN ROUTINE
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.ATTN ('ATTENTION' PROCESSOR)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS A DRIVE ATTENTION CONDITION INTERNAL
                                ; PACKET (OP CODE EQUALS 'OP.ATT'). A GET STATUS COMMAND WILL BE
                                ; ISSUED TO DETERMINE THE ATTENTION CONDITION. ONCE THE DRIVE'S
                                ; STATUS HAS BEEN DETERMINED APPROPRIATE STATUS CHANGES WILL BE MADE
                                ; OR ADDITIONAL SDI COMMANDS WILL BE ISSUED TO CLEAR UP THE ATTENTION
                                ; CONDITION. FINALLY A DRIVE CLEAR COMMAND WILL ALWAYS BE ISSUED TO
                                ; CLEAR UP ANY DRIVE ERROR CONDITIONS.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       CLEARED DRIVE ATTENTION CONDITION
                                ;       UPDATED SDI STATUS
                                ;       REGISTERS R11,R12,R13,R14,R16 AND R17 ARE USED BY THIS
                                ;       ROUTINE AND ITS SUBORDINATE ROUTINES
                                ;
                                ;;-

001053  033751  000255  111166  D.ATTN: MOV     #V.ATTN&LOBYT,R11 %JMP  D.VECP  ; R11=LO VECTOR TABLE ADDR/GO START VECTOR PROCESS
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.AVAL (D.PROC AVAILABLE FUNCTION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SERVE AS THE BASE PROCESS WHICH WILL VECTOR
                                ; INTO THE VECTOR TABLE AT OFFSET 'V.AVAL' IN ORDER TO PERFORM THE
                                ; SDI LEVEL TWO COMMANDS NECESSARY TO ACCOMPLISH THE MSCP AVAILABLE
                                ; COMMAND WITH AND WITHOUT THE 'SPIN DOWN' MODIFIER.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       SDI LEVEL TWO SPIN DOWN COMMAND ISSUED
                                ;       DRIVE DISCONNECTED
                                ;
                                ;;-

001054  033751  000321  101166  D.AVAL: MOV     #V.AVAL&LOBYT,R11 %JMP  D.VECP  ; R11=LO VECTOR TABLE ADDR/GO INIT STATE VECTOR
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.ONLN (D.PROC ONLINE FUNCTION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SERVE AS THE BASE PROCESS WHICH WILL VECTOR
                                ; INTO THE VECTOR TABLE AT OFFSET 'V.ONLN' IN ORDER TO PERFORM THE
                                ; SDI LEVEL TWO COMMANDS NECESSARY TO ACCOMPLISH THE MSCP ONLINE COMMAND.
                                ;       U.ONLN SETS SEEK WHICH HAS PRIORITY OVER THIS LEVEL OF PROCESSING
                                ; D.SEEK ACTUALLY BRINGS THE DRIVE ONLINE THROUGH CODE AT S.SEEK.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       UNIT CHARACTERISTICS OBTAINED
                                ;       DRIVE ONLINE
                                ;
                                ;;-

001055  010555  007003  123604  D.ONLN: ADD     #SDI.SW,DBAR\N,BAR %CALL S.LD12 ; R12=CURRENT ONLINE STATUS
001056  014552  000017  000000          BIT     #<ST.MFE!ST.DRV>,R12            ; IF NOT ON DISCONNECT CYCLE
001057  033751  010311  111166          MOV     #V.ONLN&LOBYT,R11 %JZRO D.VECP  ;   THEN ONLINE/R11=LO VECTOR TABLE ADDR/GO INIT STATE VEC
001060  033751  000324  101166          MOV     #V.ONAV&LOBYT,R11 %JMP  D.VECP  ;   ELSE DO AVAILABLE WITHOUT SETTING "S" BITS
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.RPLC (PROCESS MSCP REPLACE COMMAND)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP REPLACE COMMAND. IT WILL
                                ; WAIT UNTIL THE DISK IS POSITIONED TO THE SECTOR PRIOR TO THE DESIRED
                                ; ONE (USING THE SECTOR AND INDEX INDICATORS FROM THE DRIVE) AND ISSUE
                                ; THE FORMAT ON SECTOR/INDEX AS APPROPRIATE.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       UPDATED HEADER CODE WITH RBN CODE AND DATA BLOCK CONTAINING
                                ;       128 COPIES OF THE RBN.
                                ;
                                ;;-

001061  013740  007005  123620  D.RPLC: MOV     #DMREG5,BAR     %CALL   S.STDB  ; SAVE SDI CNTL BLK PTR FOR LATER OR TIMEOUT
001062  013440  000400  122475           NOP             @SDPF   %CALL   D.BFCK  ; GO LOAD HDR & DATA PREAMBLES
001063  133544  000200  122467           BIS     #PLOCK,RLL      %CALL   D.IOPR  ; RELEASE U.PROC/SET UP HEADER SEARCH LIMIT
                        ; ***   CLEAR HEADER CODE OF BUF.HH IN CASE ERROR RECOVERY ***
001064  010557  007004  133603           ADD     #BUF.HH,R17\N,BAR %CALL S.LD11  ; R11=HI LBN TO REPLACE
001065  135551  003360  010000           BIC     #HDCOD,R11,BUF                  ; CLEAR HEADER CODE & RESET
                        ; ***   SELECT CORRECT TRACK NUMBER (GROUP SELECTED BY FORCED SEEK) ***
001066  033751  000060  132507           MOV     #<IDX+SEC>,R11  %CALL   D.CPUL  ; WAIT FOR SECTOR/INDEX PULSE
001067  013740  007260  133623  DRPLCA: MOV     #REPSTA,BAR     %CALL   S.CLRB  ; [EERREC]CLEAR REPLACE STATUS WORD
001070  010557  007005  000000           ADD     #BUF.TA,R17\N,BAR               ; [16K]BAR=PTR TO R/T COMMAND
001071  033751  000010  122513           MOV     #8.,R11         %CALL   D.WSIG  ; [16K]GO WAIT FOR RCVR RDY & R/W RDY
001072  033712  000523  010000           MOV     (BUF),R12 @SWM                  ; [16K]R12=RTC,SET WRITE MODE
001073  013440  000440  123014           NOP             @SCMD   %CALL   HDRCMP  ; [16K]SET CMD MODE/GO SELECT CORRECT TRACK
001074  013440  000500  133064           NOP             @RSE    %CALL   HDRANL  ; GO VERIFY CYL/TRACK POSITION
001075  112551  010005  111067           CMP     #5,R11          %JZRO   DRPLCA  ; LOOP UNTIL WE'VE READ A VALID HEADER
001076  133754  010120  002336           MOV     #RBNPRM,R14     %JNZRO  D.TKER  ; IF HARD I/O ERR OR OFF TRACK, ERROR
                        ; ***   PUT THE RIGHT HEADER TYPE CODE ON THIS SECTOR ***
001077  010555  007022  123610           ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD16 ; GET POINTER TO MSCP PACKET
001100  010556  007007  123603           ADD     #P.MOD,R16\N,BAR %CALL  S.LD11  ; GET MSCP MODIFIER FLAGS
001101                                   ASSUME  MD.PRI,EQ,BIT00                 ; CHECK TYPE OF REVECTOR
001101  133774  040060  151102           MOV\T   #RBN2ND,R14     %TNLSB          ; IF SECONDARY, MODIFY HEADER TYPE
001102  010557  007004  010000           ADD     #BUF.HH,R17\N,BAR               ; BAR=POINTER TO HI HEADER
001103  036514  000003  133617           XOR     (BUF),R14       %CALL   S.ST14  ; STORE BACK PRIMARY OR SECONDARY REVECT HDR
                        ; ***   OBTAIN CORRECT DISK ROTATIONAL POSITION ***
001104  033752  000001  000000           MOV     #1,R12                          ; set retry count to 1          [C118]
                        DRPLCX:                                                 ; retry entry point             [C118]
001105  033751  000040  122507           MOV     #IDX,R11        %CALL   D.CPUL  ; WAIT FOR INDEX PULSE
001106  010556  007031  123605           ADD     #S.SECI,R16\N,BAR %CALL S.LD13  ; R13=SECTOR'S FROM INDEX
001107  013440  010000  101112  DRPLCB: NOP                     %JZRO   DRPLCC  ; IF SECTOR EQ 0 THEN DO IT     [C118]
001110  033751  000020  122507           MOV     #SEC,R11        %CALL   D.CPUL  ; WAIT FOR SECTOR PULSE
001111  031453  000013  101107           DEC     R13             %JMP    DRPLCB  ; DECR LOOP COUNT AND LOOP
                        DRPLCC:
001112  031452  140012  011120           DEC     R12             %JNDSER 10$     ; If pulse/parity error during  [C118]
                                                                                ; counting of sectors, dec retry[C118]
001113  033771  030001  141116           MOV\T   #1,R11          %JNEG   5$      ; count, and retry if not done  [C118]
```



```
001114  013440  000460  010000           NOP             @SCLR                   ; TO RETRY:clear error          [C118]
001115  013440  000660  111105           NOP             @RCLR   %JMP    DRPLCX  ;  and try again                [C118]
                        5$:                                                     ;                               [C118]
                                                                                ; otherwise set                 [C118]
                                                                                ; error, we have lost our place [C118]
001116  013740  007260  123614           MOV     #REPSTA,BAR     %CALL   S.ST11  ; save error                    [C118]
001117  013440  000000  111127           NOP                     %JMP    DRPLCD  ; and skip replace              [C118]
                        10$:
001120  133752  000115  000000           MOV     #FORSCD,R12                     ; set level 1 command           [C118]
                        ; ***   SET UP TO USE SPECIAL ENTRY INTO XFC01 ROUTINE ***
001121  010557  007002  123607           ADD     #BUF.BP,R17\N,BAR %CALL S.LDDB  ; DBAR=BUFFER POINTER
001122  033753  007007  000000           MOV     #DMREGO,R13,BAR                 ; R13,BAR=PTR TO DM REG 0
001123  133553  003200  133053           BIS     #BIT15,R13,BUF  %CALL   D.LDHD  ; SET TO FORMAT 1 SECTOR/LOAD HEADERS
001124  033757  000060  010000           MOV     #<IDX+SEC>,R17                  ; R17=SECTOR+INDEX FOR CHK IN XFC1A
001125  133756  000001  000000           MOV     #SECSZ,R16                      ; [ECO#1]R16=SECTOR SIZE
001126  130456  000016  133405           INC     R16             %CALL   XF.REP  ; [16K]INCR R16 FOR EDC/GO FORMAT SECTOR
001127  013740  007005  123607  DRPLCD: MOV     #DMREG5,BAR     %CALL   S.LDDB  ; RESTORE DBAR
001130  033752  000240  010000           MOV     #<SUSP+XCMP>,R12                ; R12=SDI STATUS TO SET
001131  013740  007260  123605           MOV     #REPSTA,BAR     %CALL   S.LD13  ; get the REPLACE status        [C118]
001132  036553  010001  151141           XOR\F   #1,R13          %JZRO   DRPLC1  ; if zero exit with success     [C118]
001133  133776  010001  141135           MOV\T   #SC.LVO,R16     %JZRO   20$     ;   if drive signal timeout     [C118]
001134  033756  000200  010000           MOV     #SC.RWR,R16                     ;or if read write ready error   [C118]
                        20$:                                                    ;                               [C118]
001135  033556  000013  000000           BIS     #ST.DRV,R16                     ; set drive error type          [C118]
001136  033751  000003  132043           MOV     #FM.SDI,R11     %CALL   D.LOGF  ; store error codes             [C118]
001137  033552  000004  010000           BIS     #DERR,R12                       ; and declare error             [C118]
001140  133552  000040  000000           BIS     #SLAT,R12                       ; ask it to be logged and exit  [C118]
001141  034453  000013  112302  DRPLC1: CLR     R13             %JMP    D.SETS  ; CLR R13/RESET STATUS/GO IDLE
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.RVCT (D.PROC REVECTORING DISPATCHER)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL DISPATCH D.PROC REVECTORING.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;       R11             SDI.ST CONTENTS
                                ;
                                ; OUTPUTS:
                                ;       CURRENT D.PROC REVECTOR STATE ENTERED
                                ;
                                ;;-

001142  010555  007000  123567  D.RVCT: ADD     #SDI.ST,DBAR\N,BAR %CALL P.LOCK ; [U52EC2]BAR=SDI PTR/LOCK U.PROC OUT
001143  016511  000003  000000          XOR     (BUF),R11\N                     ; [U52EC2]CHECK IF SDI STATUS HAS CHANGED
001144  037756  010134  012305          COM     #<-V.DRVC-1>&377,R16 %JNZRO D.IOEX ; [UQA][U52EC2]IF CHANGED THEN POTENTIAL RACES
                                                                                ; INVOLVING UPPER REVECTOR CODE SETTING
                                                                                ; SEEK+SUSP AND THEN INCREMENTING THE
                                                                                ; DISPATCH POINTER; DPROC SEES OLD
                                                                                ; STATUS AND NEW DISPATCH PTR AND AWAY WE GO....
                                                                                ; [U52EC2]ALSO RACE BETWEEN RVCT CLR IN UPPER
                                                                                ; [U52EC2]AND RVCSDI SET IN THIS CODE !!!
                                                                                ; [U52EC2]  ELSE R16=PTR TO START OF VECTOR AREA
001145  013740  007232  123604          MOV     #RVCSDI,BAR     %CALL   S.LD12  ; GET SDI CURRENTLY BEING REVECTORED IN R12
001146  033452  013015  051151          MOV\F   DBAR,R12,BUF    %JNZRO  D.RVCA  ; [U52EC1]IF ACTIVE THEN CONTINUE
001147  013740  007252  133621          MOV     #RVECDP,BAR     %CALL   S.ST16  ; [UQA][U52EC1]  ELSE SET START VECTOR ADDRESS
001150  013740  007254  133624          MOV     #RVCFLG,BAR     %CALL   INIT1   ; [UQA]INIT FLAG SO U.PROC GOES FIRST
001151  010555  007100  123611  D.RVCA: ADD     #SDI.PH,DBAR\N,BAR %CALL S.LD17 ; [V05][U52EC2][U52EC1]R17=POINTER TO REVECTOR BUFFER IN C
001152  013740  007254  123610          MOV     #RVCFLG,BAR     %CALL   S.LD16  ; [UQA]R16=U.PROC/D.PROC FLAG
001153  112152  040015  012305          CMP     DBAR,R12        %JLSB   D.IOEX  ; [UQA][U52EC2]IGNORE IF IT'S UPROC'S TURN
001154  013740  017226  012305          MOV     #RVCBUF,BAR     %JNZRO  D.IOEX  ; [U52EC2]IF NOT THIS SDI THEN GO IDLE
001155  133544  000200  010000          BIS     #PLOCK,RLL                      ; [U52EC2]  ELSE RELEASE U.PROC FROM LOCK
001156  013740  007252  123610          MOV     #RVECDP,BAR     %CALL   S.LD16  ; [UQA]GET DISPATCH VECTOR INTO R16
001157  013740  007254  133605          MOV     #RVCFLG,BAR     %CALL   S.LD13  ; [UQA]LOAD FLAG INTO R11
001160  037776  030131  041161          COM\T   #<-V.RVC8-1>&LOBYT,R16  %TNMSB  ; [UQA]IF NO MSB, SET FOR CLEAN UP
001161  013740  007226  010000          MOV     #RVCBUF,BAR                     ; [UQA]BAR=RVCBUF FOR REVECT ROUTINE
001162  120356  002016  010000          INCB    R16\0,R16,PAR                   ; DISPATCH TO ROUTINE & INC VECTOR
                                                                                ; AT DISPATCH;
                                                                                ; R11=SDI.ST
                                                                                ; BAR=RVCBUF
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.SEEK
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL INITIATE A SEEK OPERATION IN RESPONSE TO THE
                                ; 'SEEK' BIT BEING SET IN THE SDI CONTROL BLOCK STATUS WORD. IT WILL
                                ; CALL D.VECP TO INITIATE THE SEEK OPERATION.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       SDI SEEK COMMAND ISSUED
                                ;
                                ;;-

001163  033751  000302  111166  D.SEEK: MOV     #V.SEEK&LOBYT,R11 %JMP  D.VECP  ; [U52EC1]R11=LO VECTOR TABLE ADDR/GO INIT STATE VECTOR
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.SUCH (D.PROC SET UNIT CHAR FUNCTION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SERVE AS THE BASE PROCESS WHICH WILL VECTOR
                                ; INTO THE VECTOR TABLE AT OFFSET 'V.SUCH' IN ORDER TO PERFORM THE
                                ; SDI LEVEL TWO COMMANDS NECESSARY TO ACCOMPLISH THE MSCP SET UNIT
                                ; CHARACTERISTICS COMMAND.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       DRIVE MODE CHANGED
                                ;
                                ;;-

001164  033751  000314  101166  D.SUCH: MOV     #V.SUCH&LOBYT,R11 %JMP  D.VECP  ; R11=LO VECTOR TABLE ADDR/GO INIT STATE VECTOR
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.TOPO (D.PROC TOPOLOGY FUNCTION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SERVE AS THE BASE PROCESS WHICH WILL VECTOR
                                ; INTO THE VECTOR TABLE AT OFFSET 'V.TOPO' IN ORDER TO PERFORM THE
                                ; SDI LEVEL TWO COMMANDS NECESSARY TO ACCOMPLISH THE MSCP TOPOLOGY COMMAND.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       DRIVE TOPOLOGY COMMAND EXECUTED
                                ;
                                ;;-

001165  033751  000251  101166  D.TOPO: MOV     #V.TOPO&LOBYT,R11 %JMP  D.VECP  ; R11=LO VECTOR TABLE ADDR/GO INIT STATE VECTOR
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.VECP  (VECTOR PROCESS INITIATE ENTRY)
                                ;       D.VECT  (VECTOR PROCESS WORKING ENTRY)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       D.VECP PROVIDES THE MECHANISM TO START OFF A VECTOR TABLE CHAIN OF
                                ; EVENTS. IT REQUIRES THAT R11 HAVE THE LO ORDER ADDRESS OF THE VECTOR TABLE
                                ; ENTRY.
                                ;       D.VECT PROVIDES THE MECHANISM FOR STEPPING THROUGH THE VECTOR TABLE
                                ; ENTRIES AND HANDLES THE TERMINATION PROCESSING ONCE THE LAST STEP HAS BEEN
                                ; EXECUTED.  IT KEYS ON THE RETURNED CONDITION CODE AND R12 FOR TERMINATION;
                                ; IF CCODE = 0 THEN THE CURRENT STEP COMPLETED ELSE IT HAS NOT
                                ; IF R12 = 0 THEN THE VECTOR PROCESS IS STILL ACTIVE AND THE NEXT STEP SHOULD
                                ; BE ACTIVATED.
                                ; IF R12 NEQ 0 THEN THE VECTOR PROCESS IS COMPLETE AND R12 CONTAINS THE BITS
                                ; TO SET IN SDI.ST.
                                ;
                                ; INPUTS:
                                ;       D.VECP          R11 = LO ORDER ADDRESS OF VECTOR TABLE START ENTRY
                                ;
                                ;       RETURN FROM     CCODE = 0 CURRENT STEP COMPLETE
                                ;       PAR DISPATCH    CCODE NEQ 0 CURRENT STEP NOT COMPLETE
                                ;                       R12 = 0 PROCEED TO NEXT STEP
                                ;                       R12 NEQ 0 VECTOR PROCESS COMPLETE, R12 CONTAINS BITS
                                ;                       TO SET IN SDI.ST, R13 CONTAINS BITS TO CLEAR
                                ;
                                ; OUTPUTS:
                                ;       VECTOR PROCESS COMPLETED
                                ;
                                ;;-

001166  010555  007021  133614  D.VECP: ADD     #SDI.SV,DBAR\N,BAR %CALL S.ST11 ; SAVE VECTOR TABLE ADDRESS
001167  013440  000000  132110  D.VECA: NOP                     %CALL   D.TIMR  ; SET SDI TIMER FOR THIS COMMAND
001170  010555  007020  123606  D.VECT: ADD     #SDI.E0,DBAR\N,BAR %CALL S.LD14 ; R14=LEVEL 0 STATE VECTOR
001171                          ASSUME  ERRINP,EQ,BIT15                 ; MAKE SURE CNMSB IS VALID
001171  010555  037021  023606          ADD     #SDI.SV,DBAR\N,BAR %CNMSB S.LD14; IF LEVL 0 NOT ACTIVE THEN R14=NORMAL STATE VECTOR
001172  010555  007022  133603          ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD11 ; R11=MSCP PKT PTR
001173  010555  007044  123604          ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD12 ; R12=1ST WORD OF SDI STATUS
001174  033453  000615  010000          MOV     DBAR,R13 @RDPF                  ; R13=SDI CONTROL BLOCK PTR
001175  010555  007045  133610          ADD     #SDI.S2,DBAR\N,BAR %CALL S.LD16 ; R16=2ND WORD OF SDI STATUS
001176  010555  007056  133611          ADD     #SDI.CW,DBAR\N,BAR %CALL S.LD17 ; R17=SDI.CW(SUBUNIT AND MASK)
001177  133554  002377  121177          OR      #HIBYT,R14,PAR  %CALL   .       ; STUFF PAR & CALL (SO WE CAN RETURN)
001200  013452  010012  010701          TST     R12             %JNZRO  D.IDLE  ; IF FAILURE THEN IDLE
001201  014552  010004  101167          BIT     #DERR,R12       %JZRO   D.VECA  ; IF R12 EQ 0 THEN CONT/DONE-R12=STATUS FLAGS TO SET
                        ; ***   VECTOR TABLE PROCESSING COMPLETE ENTRY ***
001202  133573  010100  041206          BIS\T   #ERRIP,R13      %JNZRO  D.VECB  ; IF ERROR THEN DONE
001203  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS
001204                          ASSUME  RVCT,EQ,BIT15                   ; MAKE SURE RVCT IS MSB
001204  114551  030100  111206          BIT     #ERRIP,R11      %JMSB   D.VECB  ; IF ERROR RECOVERY IN PROGRESS
001205  013440  010000  010776          NOP                     %JNZRO  D.ER1A  ;   THEN GO FINISH IT
                        ; ***   R12=SDI STATUS BITS TO SET,R13=BITS TO ZAP,RESET STATUS, GO IDLE ***
001206  033553  000010  112302  D.VECB: BIS     #VECT,R13       %JMP    D.SETS  ; RESET STATUS/GO IDLE
                        .PAGE
```



```
                                .SBTTL  SDI LEVEL 0,1, AND 2 ROUTINES

                                ; *-------------------------------------------------------------------------*
                                ; *                       SDI LEVEL 0 ROUTINES                              *
                                ; *-------------------------------------------------------------------------*
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEVOWR
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE SDI LEVEL 0 CONTROL INFORMATION TRANSFER
                                ; FROM THE CONTROLLER TO THE DRIVE AS DESCRIBED IN SDI SPECIFICATION.  THE
                                ; CONTROL INFORMATION FRAME IS IN R12 AND R11 IS USED AS A SUCCESSFUL OR
                                ; UNSUCCESSFUL INDICATOR.
                                ;
                                ; INPUTS:
                                ;       R12             CONTROL INFORMATION CODE, MUST BE NONZERO
                                ;
                                ; OUTPUTS:
                                ;       R11             0 = SUCCESSFUL
                                ;                       1 = DRIVE DATA OR DRIVE CLOCK TIMEOUT
                                ;       R12             0 IF R11 = 0 (I.E. SUCCESS)
                                ;
                                ;;-

                                ; ***   ENTRY POINT FOR CONTINUE AND END FRAMES (WAIT FOR CONTROLLER TO SEND FRAME) ***
001207  033751  000020  131270  LEVOWW: MOV     #16.,R11        %CALL   D.CLCS  ; R11=FRAME SENT COUNTER/ZAP RECVD BIT
001210  013440  010000  101212  LEVOWX: NOP                     %JZRO   LEVOWR  ; IF R11 EQ 0 THEN DRIVE ON/ELSE R11=DRIVE STATE
001211  031451  110011  011210          DEC     R11             %JNCSR  LEVOWX  ; WAIT FOR FRAME SENT
                        ; ***   STANDARD ENTRY POINT ***
001212  037151  140531  131271  LEVOWR: XNOR    R11,R11 @SWM    %CDSER  D.RCLR  ; [ECO#2]R11=1'S/IF PARITY ERR THEN CLEAR
001213  031311  010442  101225  LEVOWA: DECB    (RTDS),R11 @SCMD %JZRO LEVOWE   ; [ECO#2]IF R11 EQ 0 THEN ERROR                  [mjt-e125]
001214                          ASSUME  DRDY,EQ,BIT00                   ; MAKE SURE DRDY IS LSB
001214  034711  040542  151213          CLR\F   (RTDS),R11 @RECC %JNLSB LEVOWA  ; [ECO#2]IF NO RECVR RDY THEN LOOP
001215  013740  047202  101225          MOV     #SYNC,BAR       %JNLSB  LEVOWE  ; [ECO#2]IF NO RECVR RDY THEN ERROR              [mjt-e125]
                                                                        ; [ECO#2]  ELSE BAR<-SYNC PTR/R11=SYNC*2
001216  013440  000700  010000          NOP             @SSE            ; [ECO#2]ENABLE SERDES
001217  013700  006003  121226          MOV     (BUF),SD        %CALL   LEVOWS  ; [ECO#2]OUTPUT SYNC TO SERDES
001220  013460  010000  161226          NOP\T                   %CZRO   LEVOWS  ; SEND OUT THE FRAME TO SERDES                   [mjt-e125]
001221  013460  010000  161226          NOP\T                   %CZRO   LEVOWS  ; SEND SOME 0'S TO SERDES                        [mjt-e125]
001222  013460  010000  161226          NOP\T                   %CZRO   LEVOWS  ; SEND MORE 0'S TO SERDES                        [mjt-e125]
001223  130471  010011  051224          INC\T   R11             %TNZRO          ; IF NOT SUCCESSFUL, R11=1                       [mjt-e125]
001224  013451  000511  127777          TST     R11     @RSE    %RET            ; TEST R11/RETURN
001225  033751  000001  127777  LEVOWE: MOV     #1,R11          %RET            ; STUFF R11=1, DON'T HAVE TO @RSE BECAUSE @SSE NEVER EXECU

001226  024472  106012  167777  LEVOWS: CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001227  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001230  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001231  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001232  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001233  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001234  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001235  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001236  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
```

```
001237  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001240  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001241  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001242  024472  106012  167777          CLR\T   R12\0,R12,SD    %RDCLK          ; IF DRV CLK TICK THEN RETURN                    [mjt-e125]
001243  013452  000512  127777          TST     R12     @RSE    %RET            ; WE DIDN'T GET DCLK!  DRIVE ISN'T LISTENING TO US.  EXIT [mj

                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEVORD
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE SDI LEVEL 0 CONTROL INFORMATION TRANSFER
                                ; FROM THE DRIVE TO THE CONTROLLER AS DESCRIBED IN THE SDI SPECIFICATION. THE
                                ; CONTROL INFORMATION READ IS RETURNED IN REGISTER R12. REGISTER R11 IS 0
                                ; FOR A SUCCESSFUL TRANSFER AND IS EQUAL TO 1 FOR A DRIVE CLOCK TIMEOUT.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       R11             0 FOR A SUCCESSFUL TRANSFER, 1 FOR A DRIVE CLOCK TIMEOUT
                                ;       R12             FRAME CODE (UPPER BYTE) AND DATA (LOWER BYTE)
                                ;
                                ;;-
                                ;       ********************************************************
                                ;       * There are some interesting timing problems in this    *
                                ;       * routine, and some subtle logic that needs elucidation.*
                                ;       *                                                      *
                                ;       * The 1 instruction loop at LEVORA: loops on the       *
                                ;       * condition NWRC.  WRC is the logical OR of:           *
                                ;       *       NOT( SERDES Word Rate Clock )                  *
                                ;       * with     ( NCRY from the ALU )                       *
                                ;       *                                                      *
                                ;       *     Therefore the 1 instruction loop at LEVORA: exits *
                                ;       * and continues at LEVOR2: on either of two conditions: *
                                ;       * the SERDES WRC goes low (ie a SYNC has been received) *
                                ;       * or NCRY is set (ie the DEC of R12 has just decremented*
                                ;       * from 0 to -1, thus the timeout count has been reached)*
                                ;       * This implies a maximum data rate of:                 *
                                ;       *       348ns/8.5bits = 40.9ns/bit = 24.4Mb/s.         *
                                ;       * for response data received within the timeout period  *
                                ;       * of 2.1ms.                                           *
                                ;       *                                                      *
                                ;       *     The 3 instruction loop at LEVORB: is only entered *
                                ;       * if WRC was seen and NCRY was not seen, and DRDY has   *
                                ;       * not been seen in 2 additional instructions, this      *
                                ;       * implies that the data rate is slower than:           *
                                ;       *       348ns*2/7.5bits = 92.8ns/bit                   *
                                ;       * Also the loop manages to transfer the data from      *
                                ;       * the SERDES within 2 instructions of DRDY going high   *
                                ;       * ( which stays high a max of 13 bits )and this implies *
                                ;       * a data rate upto:                                    *
                                ;       *       348ns*2/13bits = 53.5ns/bit                    *
                                ;       * can be handled. Therefore this loop is okay.         *
                                ;       *                                                      *
                                ;       *    If the initial timeout count is decremented to -1  *
                                ;       * then at LEVORB: the rcvr rdy signal is turned off. But*
                                ;       * it will take 23 bit times plus 2us before we can      *
                                ;       * guarantee that the drive has seen it during which time*
                                ;       * the controller must still listen for the command.  It *
```

```
                                ;       * is now to late to check WRC as at high data rates it  *
                                ;       * may have already been missed.  So the loop at LEVORE  *
                                ;       * is used to timeout the receiver ready race condition. *
                                ;       * It will dump the SERDES data within 2 instructions of *
                                ;       * DRDY being asserted, and this implies that the max   *
                                ;       * data rate that can be handled by this loop is:       *
                                ;       *       348ns*2/13bits = 53.5ns/bit = 18.67Mbits/sec    *
                                ;       *                                                       *
                                ;       * Therefore for responses from drives with a data rate  *
                                ;       * in excess of 18.6 Mbits/sec there is a possiblity of  *
                                ;       * missing the data if it is not received within 2.1ms   *
                                ;       * of the controller bringing up RCVR RDY.  This can     *
                                ;       * happen on commands such as:                           *
                                ;       *       DIAGNOSE                                        *
                                ;       *       DRIVE CLEAR                                     *
                                ;       *       RUN                                             *
                                ;       *       TOPOLOGY                                        *
                                ;       *******************************************************
001244  133751  000030  000000  LEVORD: MOV     #6144.,R11                      ; R11=APPROX 2 MSECS VALUE
001245  013440  000640  010000          NOP             @RCMD                   ; RESET COMMAND MODE
001246  031452  000731  000000          DEC     R11,R12 @SRM                    ; SET CARRY/SET R12/SET READ MODE
001247  013740  005011  000000          MOV     #<CONT+CRDY>,RTCS               ; SET CONTROLLER RECVR RDY
001250  031472  170712  051250  LEVORA: DEC\T    R12     @SSE    %JNWRC  LEVORA  ; DECR R12/IF NOT WRC OR CRY THEN LOOP
001251  033732  100006  151257  LEVOR2: MOV\T    (SD),R12        %JDCLK  LEVORC  ; IF DRIVE TICK THEN R12=SERDES DATA
001252  033732  100006  151257          MOV\T    (SD),R12        %JDCLK  LEVORC  ; IF DRIVE TICK THEN R12=SERDES DATA
001253                                  ASSUME  CONT,EQ,1                       ; CONT MUST BE 1 !
001253  033771  025001  041261  LEVORB: MOV\T    #CONT,R11,RTCS  %JNCRY  LEVORE  ; [ECO#2]R11=DRIVE CLOCK FAILURE ERROR
001254  033732  100006  151257          MOV\T    (SD),R12        %JDCLK  LEVORC  ; IF DRIVE TICK THEN R12=SERDES DATA
001255  031451  100011  011253          DEC     R11             %JNDCLK LEVORB  ; [ECO#2]DECR R11/IF NO DCLK THEN GO CHK FOR NCRY
001256  033712  000006  000000          MOV     (SD),R12                        ; [ECO#2]R12=SERDES DATA
                                ; ***   SDI LEVEL 0 READ SUCCESS EXIT POINT ***
001257  013740  005001  010000  LEVORC: MOV     #CONT,RTCS                      ; [ECO#2]RESET CONTROLLER RCVR RDY
001260  034451  000511  111266  LEVORX: CLR     R11     @RSE    %JMP    LEVORG  ; [ECO#2]RESET SERDES ENABLE

                                ; ***   LISTEN TO DRIVE AFTER RECEIVER READY RESET ***
001261  033732  100006  141260  LEVORE: MOV\T    (SD),R12        %JDCLK  LEVORX  ; [ECO#2]IF DRV CLK THEN DONE
001262  033752  000036  010000          MOV     #30.,R12                        ; [ECO#2]R12=LOOP COUNT
001263  033732  100006  141260  LEVORF: MOV\T    (SD),R12        %JDCLK  LEVORX  ; [ECO#2]IF DRV CLK THEN DONE
001264  031452  100012  101263          DEC     R12             %JDCLK  LEVORF  ; [ECO#2]IF DRV CLK THEN DONE ELSE DECR R12
001265  013440  010500  041263          NOP\F           @RSE    %JNZRO  LEVORF  ; [ECO#2]IF NEQ 0 THEN LOOP ELSE DONE
                                ; ***   SDI LEVEL 0 READ EXIT POINT ***
001266  013440  000440  000000  LEVORG: NOP             @SCMD                   ; SET IN COMMAND MODE
001267  013451  000531  137777          TST     R11     @RRM    %RET            ; TEST R11/RESET READ MODE/RETURN

001270  013740  005001  137777  D.CLCS: MOV     #CONT,RTCS      %RET            ; RESET ALL CONTROLLER STATE BITS

                                ; ***   ROUTINE TO RESET RTDS PARITY ERROR [ECO#2] ***
001271  013440  000460  010000  D.RCLR: NOP             @SCLR                   ; [ECO#2]SET CLEAR
001272  013311  000662  137777          TSTB    (RTDS),R11 @RCLR %RET           ; [ECO#2]RESET CLEAR/PUT LSB OF RTDS ON CCODE
                                                                                ; [ECO#2]AND R11 IN CCODE/RETURN

                                        .PAGE
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *                       SDI LEVEL 1 COMMANDS/ROUTINES                     *
                                ; *-------------------------------------------------------------------------*
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEV1RD
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL READ THE SDI INTERCONNECT FOR A DRIVE RESPONSE.
                                ; ON A SUCCESSFUL RETURN R12 WILL HOLD THE SDI DRIVE RESPONSE OP CODE AND
                                ; BUFFER POINTED TO BY R13 WILL HOLD THE DATA (IF ANY) ASSOCIATED WITH THAT
                                ; RESPONSE. R11 WILL INDICATE THE STATUS OF THE OPERATION. THIS ROUTINE WILL
                                ; ALSO VERIFY THAT THE FRAMING CODE SEQUENCE AND THE CHECKSUM ARE CORRECT.
                                ;       R17 IS A 2-BYTE SHIFT REGISTER WHICH IS DUMPED INTO MEMORY ON EVEN BYTES.
                                ;       END FRAMES ARE PROCESSED AS NORMAL FRAMES AS MUCH AS POSSIBLE TO ALLOW
                                ;       THEM TO CLEAR OUT THE R17 SHIFT REGISTER.
                                ;
                                ; INPUTS:
                                ;       R13             POINTER TO DRIVE RESPONSE BUFFER
                                ;       R14             LENGTH OF DRIVE RESPONSE BUFFER
                                ;
                                ; OUTPUTS:
                                ;       R11 & LV1SV1    0 = SUCCESSFUL
                                ;               DCERR - 1  = DRIVE CLOCK OR DRIVE DATA TIMEOUT
                                ;               SFERR - 2  = FIRST WORD NOT A START FRAME
                                ;               FRERR - 4  = ILLEGAL FRAME CODE
                                ;               CSERR - 10 = INVALID CHECKSUM
                                ;               LNERR - 20 = RESPONSE LENGTH ERROR
                                ;
                                ;       R12/LV1SV2      OP CODE OF RESPONSE PACKET
                                ;       R14             0 = DRIVE BUFFER NOT OVERRUN
                                ;                       NEGATIVE = DRIVE BUFFER OVERRUN
                                ;       R16 IS USED TO CALCULATE THE CHECKSUM
                                ;       R17 IS USED AS A TEMPORARY REGISTER
                                ;;-
001273  013740  007212  121244  LEV1RD: MOV     #LV1SV1,BAR     %CALL   LEVORD  ; GET START FRAME
001274  033456  010012  077777          MOV\F    R12,R16         %RNZRO          ; RETURN IF TROUBLE
001275  034556  000377  133623          AND     #LOBYT,R16      %CALL   S.CLRB  ; ISOLATE OPCODE, CLEAR ERROR BYTE
001276  013740  007214  000000          MOV     #LV1SV2,BAR                     ; [16K]BAR=PTR TO OP CODE SAVE
001277  035552  000377  123621          BIC     #LOBYT,R12      %CALL   S.ST16  ; ISOLATE FRAME CODE, SAVE OPCODE
001300  116552  000161  000000          XOR     #STRTCD,R12\N                   ; CHECK FOR START FRAME
001301  033771  010002  071330          MOV\T    #SFERR,R11      %CNZRO  LEV1RF  ; ANYTHING ELSE IS ERROR
001302  130154  010014  011304          ADDC    R14,R14         %JNZRO  LEV1RB  ; BUT KEEP GOING SINCE IT MAY BE END FRAME
                                                                                ; NOTE THAT R14 IS NOW A BYTE COUNT + 1
001303  035557  000377  121244  LEV1RA: BIC     #LOBYT,R17      %CALL   LEVORD  ; CLEAR NEW BYTE IN R17 AND GET NEW BYTE
001304  030156  010012  001330  LEV1RB: ADD     R12,R16         %JNZRO  LEV1RF  ; ACCUMULATE CHECKSUM IN R16
001305                                  ASSUME  CONTCD&400,EQ,0                 ; SINCE THE LEGAL FRAMES HAVE BIT 8 CLEAR
001305                                  ASSUME  ENDCD&400,EQ,0                  ; WE CAN USE IT FOR A CARRY BIT...
001305  114556  000001  010000          BIT     #400,R16                        ; DO END AROUND CARRY, BUT
001306  130476  010016  051307          INC\T    R16             %TNZRO          ; LEAVE UPPER HALF SLOPPY FOR NOW...
001307  036157  000012  010000          XOR     R12,R17                         ; R17(LO) = DATA BYTE
001310  035552  000377  000000          BIC     #LOBYT,R12                      ; ISOLATE FRAME CODE IN R12
```

```
001311  036157  000012  010000          XOR     R12,R17                         ; R17(HI) IS BACK TO ITS OLD VALUE
001312  033751  000010  123577          MOV     #8.,R11         %CALL   S.RR17  ; SWAP BYTES OF R17
001313  033751  000004  000000          MOV     #FRERR,R11                      ; PREPARE TO CHECK FRAME CODE
001314  116552  000324  000000          XOR     #CONTCD,R12\N                   ; CHECK CONTINUE
001315  116552  010262  111317          XOR     #ENDCD,R12\N    %JZRO   LEV1RC  ; OR END FRAME
001316  135557  010377  071330          BIC\F   #HIBYT,R17      %CNZRO  LEV1RF  ; FORCE R17(HI) TO 0 IF END FRAME
001317  031454  000014  000000  LEV1RC: DEC     R14                             ; DECREMENT BUFFER BYTE COUNT
001320                          ASSUME  DCERR,EQ,BIT00                          ; MAKE SURE DCERR IS LSB
001320  033771  030020  171330          MOV\T   #LNERR,R11      %CMSB   LEV1RF  ; SET LENGTH ERROR
001321  120373  047013  073622  LEV1RE: INCB\T   R13\0,R13,BAR   %CLSB   S.ST17  ; IF POSITIVE AND ODD, STORE WORD AND INC PTR
001322  136552  000262  010000          XOR     #ENDCD,R12                      ; SEPARATE END FRAMES FROM CONTINUE FRAMES
001323  034576  010377  041303          AND\T   #LOBYT,R16      %JNZRO  LEV1RA  ; MAKE CHECKSUM NEAT AND LOOP UNLESS END FRAME

                                ;       NOTE THAT WE DO NOT CLEAN UP R16 ON END FRAMES - THIS IS TO CATCH THE NITPICKING
                                ;       SPECIAL CASE OF   377+0 = 377+377 (WITH END-AROUND CARRY)

001324  137556  000115  010000          XNOR    #<-ENDCD-1>&HIBYT,R16           ; COMPARE CHECKSUM AGAINST EXPECTED VALUE,
                                                                                ; WHICH IS ENDCD+377.
001325  033751  010010  031330          MOV     #CSERR,R11      %CNZRO  LEV1RF  ; LOG CHECKSUM ERROR IF IT AIN'T RIGHT
001326  013740  007214  133604          MOV     #LV1SV2,BAR     %CALL   S.LD12  ; [16K]GET OPCODE IN R12 FOR EXIT
001327  013740  007212  113603          MOV     #LV1SV1,BAR     %JMP    S.LD11  ; [16K]GET ERROR CODE IN R11 AND EXIT
001330  013740  007212  000000  LEV1RF: MOV     #LV1SV1,BAR                     ; [16K]ROUTINE TO "OR" ERROR CODE IN R11
001331  033511  000003  103614          BIS     (BUF),R11       %JMP    S.ST11  ; WITH CUMULATIVE CODE IN HEADER
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       STRT.F
                                ;       CONT.F
                                ;       END.F
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES FORM THE SDI LEVEL 1 COMMANDS AS  DESCRIBED IN
                                ; THE CONTROLLER FUNCTIONAL SPECIFICATION SECTION 5.5.4. THE ROUTINES ARE DESCRIBED
                                ; AS FOLLOWS:
                                ;       STRT.F          OUTPUTS MESSAGE START FRAME
                                ;       CONT.F          OUTPUTS MESSAGE CONTINUATION FRAME
                                ;       END.F           OUTPUTS MESSAGE END FRAME
                                ;
                                ; INPUTS:
                                ;       R12             DATA FOR STRT.F (UPPER BYTE = 0)
                                ;                       DATA FOR CONT.F (UPPER BYTE = 0)
                                ;                       CHECKSUM FOR END.F
                                ;       R13             COPY OF DATA FOR STRT.F
                                ;
                                ; OUTPUTS:
                                ;       R11             0 = SUCCESSFUL COMPLETION
                                ;                       1 = DRIVE DATA OR DRIVE CLOCK TIMEOUT
                                ;       R13             CHECKSUM (STARTED BY STRT.F,MAINTAINED BY
                                ;                       CONT.F,AND OUTPUT TO DRIVE BY END.F)
                                ;
                                ;;-

                                ; ***   OUTPUT DATA AND START FRAME CODE ***

001332  133552  000161  101212  STRT.F: BIS     #STRTCD,R12     %JMP    LEV0WR  ; GO TO LEVEL 0 TO TRANSMIT/RETURN

                                ; ***   OUTPUT DATA AND CONTINUE FRAME CODE ***

001333  030153  000012  000000  CONT.F: ADD     R12,R13                         ; UPDATE CHECKSUM
001334  114553  000001  010000          BIT     #400,R13                        ; CHECK FOR BYTE CARRY
001335  131573  010377  051336          SUB\T   #377,R13        %TNZRO          ; IF CARRY THEN ADD 1 TO CHECKSUM
001336  133552  000324  111207          BIS     #CONTCD,R12     %JMP    LEV0WW  ; GO TO LEVEL 0 TO TRANSMIT/RETURN

                                ; ***   OUTPUT CHECKSUM AND END FRAME CODE ***

001337  037452  000013  010000  END.F:  COM     R13,R12                         ; 1'S COMPLEMENT CHECKSUM
001340  136552  000115  111207          XOR     #NENDCD,R12     %JMP    LEV0WW  ; GO TO LEVEL 0 TO TRANSMIT/RETURN
                                        .PAGE
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *                    SDI LEVEL 2 COMMANDS/ROUTINES                        *
                                ; *-------------------------------------------------------------------------*
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEV2.1
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL ISSUE THE 1 BYTE LEVEL TWO COMMAND THAT IS PASSED
                                ; IN R12.
                                ; THE SDI COMMAND RESPONSE TIMER WILL BE SET PRIOR TO SENDING THE COMMAND
                                ; TO THE DRIVE.
                                ;
                                ; INPUTS:
                                ;       R12             SDI LEVEL 2 COMMAND
                                ;
                                ; OUTPUTS:
                                ;       R11             0 = SUCCESSFUL COMPLETION
                                ;                       1 = DRIVE DATA OR DRIVE CLOCK TIMEOUT
                                ;       R12 IS USED AS A TEMPORARY REGISTER
                                ;       R13 IS USED FOR THE SDI LEVEL 1 CHECKSUM
                                ;
                                ;;-

001341  033453  000012  121332  LEV2.1: MOV     R12,R13         %CALL   STRT.F  ; R12=OP CODE/OUTPUT START FRAME
001342  013451  000011  111345          TST     R11             %JMP    LEV2.E  ; TEST R11/CONTINUE
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEV2.2
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL ISSUE A TWO BYTE LEVEL TWO COMMAND USING THE
                                ; COMMAND PASSED IN R12 AND THE DATA PASSED IN R14.
                                ; THE SDI COMMAND RESPONSE TIMER WILL BE SET PRIOR TO SENDING THE COMMAND
                                ; TO THE DRIVE.
                                ;
                                ; INPUTS:
                                ;       R12             SDI LEVEL 2 COMMAND
                                ;       R17             DATA FOR SDI LEVEL 2 COMMAND IN R12
                                ;
                                ; OUTPUTS:
                                ;       R11             0 = SUCCESSFUL COMPLETION
                                ;                       1 = DRIVE DATA OR DRIVE CLOCK TIMEOUT
                                ;       R12 IS USED AS A TEMPORARY REGISTER
                                ;       R13 IS USED FOR THE SDI LEVEL 1 CHECKSUM
                                ;
                                ;;-

001343  033453  000012  121332  LEV2.2: MOV     R12,R13         %CALL   STRT.F  ; R12=COMMAND TO SEND
001344  033472  010017  171333          MOV\T   R17,R12         %CZRO   CONT.F  ;   ELSE OUTPUT DATA BYTE
001345  013440  010000  131337  LEV2.E: NOP                     %CZRO   END.F   ;   ELSE OUTPUT END FRAME
001346  013451  000011  111714          TST     R11             %JMP    D.ECKS  ; GO CHK FOR SEND ERRORS
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       LEV2.T (TABLE DRIVEN)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SEND THE SDI LEVEL TWO COMMAND THAT IS THE SECOND
                                ; (HIGH) BYTE IN THE TABLE POINTED TO BY REGISTER R16. IT WILL TRANSFER THE
                                ; NUMBER OF BYTES (INCLUDING COMMAND) INDICATED BY REGISTER R14.
                                ;
                                ; INPUTS:
                                ;       R16             POINTER TO SDI COMMAND TABLE (BUFFER)
                                ;       R17             NUMBER OF BYTES IN COMMAND (INCLUDING COMMAND BYTE)
                                ;
                                ; OUTPUTS:
                                ;       R11             0 = SUCCESSFUL COMPLETION
                                ;                       1 = DRIVE DATA OR DRIVE CLOCK TIMEOUT
                                ;       R12 IS USED AS A TEMPORARY REGISTER
                                ;       R13 IS USED FOR THE SDI LEVEL 1 CHECKSUM
                                ;       R17 IS USED AS A TEMPORARY REGISTER
                                ;
                                ;;-
001347  120156  007016  133604  LEV2.T: ADDC    R16\0,R16,BAR   %CALL   S.LD12  ; R12=SDI COMMAND(HI)
001350  035552  000377  133575          BIC     #LOBYT,R12      %CALL   S.SWAB  ; SWAP BYTES OF R12
001351  033453  000012  121332          MOV     R12,R13         %CALL   STRT.F  ; OUTPUT START FRAME AND COMMAND
001352  150456  010016  077777  LEV2TA: INC\RF   R16             %RNZRO          ; IF ERROR THEN RET/ELSE INCR R16
001353  031457  000017  000000          DEC     R17,R17                         ; DEC R17
001354  073456  017016  141337          MOV\LF  R16,R16,BAR     %JZRO   END.F   ; IF R17 EQ 0 THEN SEND END FRAME
001355  033712  030003  133575          MOV     (BUF),R12       %CMSB   S.SWAB  ; R12=2 DATA BYTES
001356  034552  000377  121333          AND     #LOBYT,R12      %CALL   CONT.F  ; OUTPUT DATA BYTE
001357  013451  000011  111352          TST     R11             %JMP    LEV2TA  ; INCR BUFFER PTR/IF BYTE CNT 0 THEN DONE
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       S.XXXX (SEND SDI LEVEL 2 COMMAND)
                                ;       R.XXXX (RECEIVE SDI LEVEL 2 RESPONSE)
                                ;
                                ; FUNCTIONAL DESCRIPTIONS:
                                ;       THESE ROUTINES WILL SEND EACH SDI LEVEL TWO COMMAND NEEDED BY THE
                                ; CONTROLLER AND ITS SISTER ROUTINE WILL RECEIVE ALL OF THE DRIVE RESPONSES. IN
                                ; ADDITION THEY WILL MAINTAIN THE STATE VECTOR INDEX 'SDI.SV' AND RETURN
                                ; SUCCESS OR FAILURE STATUS TO THE CALLING ROUTINE. CERTAIN COMMANDS
                                ; WILL ALSO DO FUNCTION SPECIFIC PROCESSING (I.E GET STATUS).
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;       R11             SDI.PQ
                                ;       R12             SDI.S1
                                ;       R13             SDI.CP
                                ;       R16             SDI.S2
                                ;       R17             SDI.CW
                                ;       BAR             ->SDI.CW
                                ;       DPF             RESET
                                ;
                                ; OUTPUTS:
                                ;       APPROPRIATE COMMAND SENT OR RESPONSE RECEIVED
                                ;       SUCCESS OR FAILURE INDICATION
                                ;
                                ;;-

                                ; ***   SEND STYLE 1 SDI CHANGE MODE COMMAND  (FOR ONLINE) ***
                                ; ***   CHECK FOR ONLINE READ RECOVERY IN PROGRESS ***
                                ;
                                ;       REMOVE WITH NEW MD.IMF HANDLING [V05]
                                ;
                                ;S.CMD1:;MOV     ##CHGMOD,R13    %JMP    S.CMD1  ; EXECUTED IN JUMP TABLE
                                ;       ADD     #P.RS06,R11,BAR %CALL   S.LD11  ; R11=RETRY WORD
                                ;       ASSUME  ONLREC,EQ,BIT15                 ; ONLINE RECOVERY FLG MUST BE MSB
                                ;       MOV     #<V.OLRD-V.CMD1>,R11 %JMSB D.INSV;  THEN SKIP LEVEL TWO COMMANDS/DO READ

                                ; ***   SEND STYLE 2 SDI CHANGE MODE COMMAND (FOR ATTENTION) ***
                                ; ***   NOW ONLY 1 STYLE SDI CHANGE MODE COMMAND [V05] ***
                                S.CMOD:;MOV     ##CHGMOD,R13    %JMP    S.CMOD  ; [V05]EXECUTED IN JUMP TABLE
                                ; ***   CHECK FOR WRITE PROTECT BIT CHANGING ***
001360  134557  000360  122023          AND     #DRV.SU,R17     %CALL   D.GUNF  ; ISOLATE SUBUNIT MASK FIELD/GO GET UNIT FLAGS
                                ; ***   CHECK FOR SOFTWARE/HARDWARE WRITE PROTECT ***
001361  114556  000060  000000          BIT     #<UF.WPH!UF.WPS>,R16            ; IF H/W OR S/W WRITE PROTECTED
001362  033152  010017  141364          BIS\F   R17,R12         %JZRO   10$     ;   THEN SET BIT TO TURN LIGHT ON        [E126]
001363  135552  000340  000000          BIC     #<DRV.W4!DRV.W3!DRV.W2>,R12     ;       CLEAR DISABLE OF ERROR LOG      [E126]
001364  134552  000374  133575  10$:    AND     #<HIBYT-DRV.DB-DRV.S7>,R12 %CALL S.SWAB ; ISOLATE HI BYTE/SWAP BYTES OF R12 [E126]
001365  133552  000367  000000          BIS     #<DRV.SU!DRV.FO!DRV.DB!DRV.S7>,R12 ; SET MASK TO CLEAR THESE BITS
001366  033552  000004  101375          BIS     #<DRV.FO/256.>,R12 %JMP S.SND3  ; ENABLE FORMATTING FOR REPLACE MD

                                ; ***   IF @RDPF THEN SEND STYLE 1 (ONLINE) SDI CHANGE FLAGS COMMAND ***
                                ; ***   IF @SDPF THEN SEND STYLE 2 (AVAILABLE) SDI CHANGE FLAGS COMMAND ***
001367  133753  000202  010000  S.CFL1: MOV     ##CHGFLG,R13                    ; R13=CHANGE FLAG CMD (HI BYTE)
```

```
001370  134557  000360  000000          AND     #DRV.SU,R17                     ; ISOLATE SUBUNIT FIELD
001371  033452  060017  011375          MOV     R17,R12         %JNDPF  S.SND3  ; R17=SUBUNIT MASK
001372  030551  007007  133575          ADD     #P.MOD,R11,BAR  %CALL   S.SWAB  ; [ECO#1]BAR=>MODIFIERS FIELD/SWAP R12
001373  033711  000003  000000          MOV     (BUF),R11                       ; [ECO#1]R11=MODIFIERS FIELD
001374                                  ASSUME  MD.SPD,EQ,BITOO                 ; IF NO SPIN DOWN
001374  033152  040017  101672          OR      R17,R12         %JNLSB  D.INS2  ; [ECO#1]THEN CHK FOR DISCONNECT
                                                                                ; [ECO#1]ELSE SET MASK AND S-BIT TO SET IN R12
001375                                  ASSUME  S.SND3,EQ,.                     ; MAKE SURE S.SND3 IS NEXT

                                ; ***   SEND THREE BYTE SDI LEVEL 2 COMMAND (R13=CMD,R12=MASK AND BITS) ***
001375  033756  007011  123616  S.SND3: MOV     #INPUT,R16,BAR  %CALL   S.ST13  ; STORE OP CODE IN PKT
001376  110440  007616  123615          INC     R16,BAR @RDPF   %CALL   S.ST12  ; STORE MASK AND BIT FLAGS
001377  033757  000003  121347          MOV     #3,R17          %CALL   LEV2.T  ; GO SEND COMMAND
001400  013451  000011  111714          TST     R11             %JMP    D.ECKS  ; GO CHECK FOR SEND ERRORS

                                ; ***   SEND SDI DRIVE CLEAR COMMAND ***
                                S.DCLR:;AND     #<LOBYT-DRV.DF>,R16 %JMP S.DCLR ; ISOLATE ERROR BYTE OTHER THAN DIAG FAIL BIT
001401  033752  010005  101672          MOV     #DRVCLR,R12     %JZRO   D.INS2  ; IF NO ERRORS THEN SKIP FRIVE CLEAR CMD
001402  033457  000016  111343          MOV     R16,R17         %JMP    LEV2.2  ; GO SEND DRIVE CLEAR COMMAND

                                ; ***   SEND SDI DISCONNECT COMMAND (DUPLICATE OR ONLINE TO CONTROLLER) ***
                                S.DCN1:;CLR     R17             %JMP    S.DCN1  ; EXECUTED IN JUMP TABLE
001403  014552  000200  010000          BIT     #DRV.OA,R12                     ; IF ALREADY ONLINE
001404  033777  010200  041413          MOV\T   #DCN.TT,R17     %JNZRO  S.DCNE  ;   THEN DISCONNECT WITH TERMINATE TOPOLOGY MOD.
001405  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS
001406  114551  000020  000000          BIT     #DRAVL,R11                      ; IF DRIVE AVAILABLE
001407  114551  010004  002074          BIT     #DRDUP,R11      %JNZRO  D.ZATT  ;   THEN DONE/IF DUPLICATE SET
001410  033777  010001  041413          MOV\T   #DCN.ST,R17     %JNZRO  S.DCNE  ;   THEN SPIN DOWN
001411  010555  007033  010000  S.DCN3: ADD     #SDI.TO,DBAR\N,BAR              ; [16K] ELSE BAR=PTR TO SDI TIMEOUT
001412  013740  003005  122045          MOV     #5,BUF          %CALL   D.OFFD  ; [16K]SET T/O=32,ALREADY ONLINE & MARK UNKNOWN
001413  033752  000204  101343  S.DCNE: MOV     #DISCON,R12     %JMP    LEV2.2  ; GO SEND DISCONNECT COMMAND

                                ; ***   SEND SDI DISCONNECT COMMAND (MSCP AVAILABLE) ***
                                S.DCN2:;MOV     R11,R16         %JMP    S.DCN2  ; EXECUTED IN JUMP TABLE
001414  010553  007060  123623          ADD     #SDI.UF,R13\N,BAR %CALL S.CLRB  ; [U52EC3]ZAP UNIT FLAGS
001415  133753  000020  122013          MOV     #DRAVL,R13      %CALL   D.SSET  ; GO MARK SDI STATUS AVAILABLE
001416  030556  007007  133611          ADD     #P.MOD,R16,BAR  %CALL   S.LD17  ; R17=MSCP MODIFIERS
001417                                  ASSUME  DCN.ST,EQ,MD.SPD                ; INSURE SPIN DOWN MODIFER COMPATIBLE
001417  114551  000014  010000          BIT     #<DRDUP!DRVOL>,R11              ; IF DRIVE WAS NOT A DUPLICATE OR OFFLINE[rae03]
001420  034557  010001  111413          AND     #DCN.ST,R17     %JZRO   S.DCNE  ;   THEN JUST DISCONNECT                [rae03]
001421  133753  000020  132011          MOV     #DRAVL,R13      %CALL   D.CLRS  ;   ELSE MARK OFFLINE                   [rae03]
001422  133753  000010  122013          MOV     #DRVOL,R13      %CALL   D.SSET  ;                                       [rae03]
001423  010555  007033  010000          ADD     #SDI.TO,DBAR\N,BAR              ; BAR=PTR TO SDI TIMEOUT                [rae03]
001424  013740  003005  000000          MOV     #5,BUF                          ; set timeout                           [rae03]
001425  033757  000001  101413          MOV     #DCN.ST,R17     %JMP    S.DCNE  ; spin down drive                       [rae03]
                                ; ***   SEND SDI ERROR RECOVERY COMMAND ***
                                S.EREC:;MOV     #ERECOV,R12     %JMP    S.EREC  ; EXECUTED IN JUMP TABLE
001426  010555  007006  133611          ADD     #SDI.UB,DBAR\N,BAR %CALL S.LD17 ; [EERREC]GET U BUFFER DESC PTR
001427  030557  007001  123611          ADD     #BUF.ST,R17,BAR %CALL   S.LD17  ; [EERREC]GET STATUS WORD
001430  133557  003001  000000          BIS     #BERDN,R17,BUF                  ; [EERREC]SET ERROR RECOVERY CMD DONE FLAG
001431  010555  007017  133611          ADD     #SDI.E1,DBAR\N,BAR %CALL S.LD17 ; R17=LEVEL 1 ERROR STATE
001432  034557  000377  101343          AND     #LOBYT,R17      %JMP    LEV2.2  ; GET RECOVERY LEVEL/GO SEND COMMAND

                                ; ***   RECEIVE GET COMMON CHARACTERISTICS RESPONSE ***
```



```
                                R.GCCH:;MOV     #SDI.DL,R14     %JMP    R.GCCH  ; EXECUTED IN JUMP TABLE
001433  033453  000015  010000          MOV     DBAR,R13                        ; R13=SDI CONTROL BLK PTR
001434  030553  000033  121273          ADD     #SDI.TO,R13     %CALL   LEV1RD  ; R13=BUF PTR/GO GET RESPONSE
001435  033756  000170  000000          MOV     #GTCRSP,R16                     ; R16=GET COMMON CHAR RESPONSE
                                ; ***   ROUTINE TO VERIFY RESPONSE IN R16 ***
001436  035551  000020  010000  R.GXCH: BIC     #LNERR,R11                      ; EXPECT A RESPONSE LENGTH ERROR
001437  013451  010011  001704          TST     R11             %JNZRO  D.ECKR  ; GO CHK FOR RECEIVE ERRORS
001440  016152  000016  101712          XOR     R16,R12\N       %JMP    D.ECRB  ; CHECK FOR CORRECT RESPONSE/CONTINUE

                                ; ***   SEND SDI GET SUBUNIT CHARACTERISTICS COMMAND ***
                                S.GUCH:;MOV     #GTUCHR,R12     %JMP    S.GUCH  ; EXECUTED IN JUMP TABLE
001441  033751  000010  123577          MOV     #8.,R11         %CALL   S.RR17  ; GET SUBUNIT MASK IN BITS <7:4>
001442  034557  000360  101343          AND     #HINIB,R17      %JMP    LEV2.2  ; GET SUBUNIT MASK/GO SEND CMD

                                ; ***   RECEIVE SDI GET SUBUNIT CHARACTERISTICS RESPONSE ***
                                R.GUCH:;MOV     #<SDI.LL-SDI.H1>,R14 %JMP R.GUCH ; EXECUTED IN JUMP TABLE
001443  030553  000063  121273          ADD     #SDI.H1,R13     %CALL   LEV1RD  ; BUMP PAST STATUS,UNIT FLAGS
001444  033756  000167  111436          MOV     #GTURSP,R16     %JMP    R.GXCH  ; R16=GET SUBUNIT RESP/GO VERIFY

                                ; ***   RECEIVE TOPOLOGY RESPONSE ***
                                R.TOPO:;MOV     #TOPRSP,R12     %JMP    R.TOPO  ; EXECUTED IN JUMP TABLE
001445                                  ASSUME  R.GSTA,EQ,.                     ; MAKE SURE RECV GET STATUS IS NEXT

                                ; ***   RECEIVE SDI GET STATUS RESPONSE ***
                                R.GSTA:;MOV     #STARSP,R12     %JMP    R.GSTA  ; EXECUTED IN JUMP TABLE
001445  013740  007010  133615          MOV     #HEADER,BAR     %CALL   S.ST12  ; [16K]SAVE RESPONSE CODE
001446  033753  000011  000000          MOV     #INPUT,R13                      ; R13=INPUT BUFFER PTR
001447  033754  000007  131273          MOV     #INPLEN,R14     %CALL   LEV1RD  ; GO TRY TO READ RESPONSE
001450                                  ASSUME  <HEADER&1>,EQ,0                 ; [16K]MAKE SURE EVEN
001450  013760  017010  151452          MOV\T   #HEADER,BAR     %JZRO   R.GSTB  ; [16K][ECO#1]NO RESPONSE GO CHK FOR RECEIVE ERRORS
001451                                  ASSUME  DCERR,EQ,1                      ; [ECO#1]MAKE SURE DCERR IS LSB
001451  033753  040001  011715          MOV     #1,R13          %JLSB   D.ERCK  ; [ECO#1]IF NO RESP THEN CHK FOR TIMEOUT
001452  016512  040003  001723  R.GSTB: XOR     (BUF),R12\N     %JLSB   D.ECKB  ; [ECO#1]CHECK RESPONSE AGAINST STORED VALUE
001453  033754  010011  001712          MOV     #INPUT,R14      %JNZRO  D.ECRB  ; IF BAD RESPONSE THEN DO ERROR RECOVERY
                                ; ***   SAVE SDI STATUS IN SDI CONTROL BLOCK ***
001454  033452  000015  000000          MOV     DBAR,R12                        ; R12=DBAR
001455  030552  000043  010000          ADD     #SDI.UN,R12                     ; R12=PTR TO START OF STATUS SAVE AREA
001456  033756  000003  122105          MOV     #3,R16          %CALL   D.SMOV  ; R16=# WDS TO MOVE/GO MOVE STATUS
001457  030552  000002  132105          ADD     #<SDI.S4-<SDI.S2+1>>,R12 %CALL D.SMOV; ADJUST R12 FOR S4->S7/GO MOVE STATUS
                                ; ***   CHECK TO SEE IF ERROR LOG SHOULD BE GENERATED ***
001460  010555  007044  133603          ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD11 ; R11=REQUEST WORD
001461  014551  000050  000000          BIT     #<DRV.DR!DRV.EL>,R11            ; IF NO DIAG OR LOG REQUEST
001462  033756  010353  112036          MOV     #<ST.DRV!SC.DDE>,R16 %JZRO D.INS1;   THEN GO TO NEXT STEP
001463  033753  000003  131633          MOV     #FM.SDI,R13     %CALL   C.LOGS  ; GO SEE IF LOGS ENABLED
001464  013440  000000  102036          NOP                     %JMP    D.INS1  ; AND GO TO NEXT STEP

                                ; ***   INITIALIZE DRIVE ***
                                S.INIT:;NOP                     %JMP    S.INIT  ; EXECUTED IN JUMP TABLE
001465  013440  000000  131022          NOP                     %CALL   D.INIT  ; INITIALIZE DRIVE
001466  013440  000000  102036          NOP                     %JMP    D.INS1  ; GO TO NEXT STATE

                                ; ***   SEND SDI ONLINE COMMAND ***
                                S.ONLN:;MOV     #SDITMO,R17     %JMP    S.ONLN  ; EXECUTED IN JUMP TABLE
001467  013440  000000  122016          NOP                     %CALL   D.DNSK  ; [U52EC2]DECR "NSEEKS"
```

```
001470  033752  000213  101343          MOV     #ONLINE,R12     %JMP    LEV2.2  ; GO SEND SDI ONLINE COMMAND

                                ; ***   SEND SDI RECALIBRATE COMMAND ***
                        S.RCAL::ADD     #SDI.S1,DBAR\N,BAR %JMP S.RCAL  ; EXECUTED IN JUMP TABLE
001471  014552  000100  010000          BIT     #DRV.RR,R12             ; IF RECALIBRATE NOT REQUESTED
001472  015552  013100  111672          BIC     #DRV.RR,R12\N,BUF %JZRO D.INS2  ;   THEN SKIP SEND & RECEIVE RECAL
001473  010555  007026  133604          ADD     #SDI.ES,DBAR\N,BAR %CALL S.LD12 ; force Seek following recalibrate      ;[E121]
001474  033552  003004  010000          BIS     #FSEEK,R12,BUF          ;                                               ;[E121]
001475  033752  000216  111506          MOV     #RECALB,R12     %JMP    S.LRSP  ; GO SEND RECALIBRATE COMMAND

                                ; ***   RECEIVE LONG TIMEOUT RESPONSE ***
                        R.LRSP::MOV     #INPUT,R13      %JMP    R.LRSP  ; EXECUTED IN JUMP TABLE
001476  013740  007200  123603          MOV     #HOSTMO,BAR     %CALL   S.LD11  ; [V05] R11 = HOST TIMEOUT
001477  013740  007174  123614          MOV     #TMR.MC,BAR     %CALL   S.ST11  ; [V05] RESET HOST TIMER
001500  010555  007035  133603          ADD     #SDI.EC,DBAR\N,BAR %CALL S.LD11 ; R11=ECC THRESHOLD WORD
001501  135571  033200  162121          BIC\T   #BIT15,R11,BUF %CMSB    D.TIML  ; IF TIMEOUT NOT SET THEN GO SET LONG TIMEOUT
001502                                  ASSUME  R.RESP,EQ,.             ;   ELSE FALL INTO R.RESP ROUTINE

                                ; ***   RECEIVE SDI LEVEL TWO RESPONSE ***
                        R.RESP::MOV     #INPUT,R13      %JMP    R.RESP  ; EXECUTED IN JUMP TABLE
001502  033754  000007  131273          MOV     #INPLEN,R14     %CALL   LEV1RD  ; R14=RESPONSE BUFR LENGTH/GO GET RESPONSE
001503  013451  000011  101704  R.RSPA: TST     R11             %JMP    D.ECKR  ; GO CHK FOR RECEIVE ERRORS

                                ; ***   SEND SDI RUN COMMAND ***
                        S.RUNN::BIC     #<DRV.AT!DRV.AV>,R17,BUF %JMP S.RUNN ; [16K]EXECUTED IN JUMP TABLE
001504  133753  000020  132011          MOV     #DRAVL,R13      %CALL   D.CLRS  ; R13=BIT TO CLR/GO CLR SDI STATUS BIT
001505  033752  000014  010000          MOV     #RUNN,R12                       ; R12=SDI RUN COMMAND/DECR NSEEKS
                                ; ***   SEND LONG RESPONSE COMMAND - R12 HAS OP CODE ***
001506  010555  007035  133603  S.LRSP: ADD     #SDI.EC,DBAR\N,BAR %CALL S.LD11 ; R11=LONG TIMEOUT FLAG WORD
001507  133551  003200  101341          BIS     #BIT15,R11,BUF  %JMP    LEV2.1  ; SET LONG TIMEOUT FLAG/SEND COMMAND

                                ; ***   SEND SDI SEEK COMMAND ***
                                ; ***   FIRST CHECK IF UNIT ONLINE ***
                        S.SEEK::MOV     DBAR,R16        %JMP    S.SEEK  ; EXECUTED IN JUMP TABLE
001510  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS
001511  114551  000020  000000          BIT     #DRAVL,R11                      ; IF DRIVE ONLINE                               ;[E125]
001512  114551  010014  111515          BIT     #<DRVOL!DRDUP>,R11 %JZRO 10$    ; THEN CHECK OTHER CONDITIONS                   ;[E125]
                                                                                ; /IF DRIVE TRUELY ONLINE(NOT OFFLINE OR DUPLICATE) [E125]
001513  037771  010014  142037          COM\T   #<V.SEEK-V.SKON-1>,R11 %JZRO D.INSV; THEN DO ONLINE COMMAND              ;[E125]
001514  013440  000000  111771          NOP                     %JMP    D.SDIG  ; ELSE GO DIE (REPORT ERROR)                    ;[E126]
001515  114551  000040  000000  10$:    BIT     #SLAT,R11                       ; IF ERROR LOG PENDING                          ;[E125]
001516  114551  010002  027777          BIT     #GSEL,R11       %RNZRO          ; THEN RETURN, ELSE IF SPECIAL SEEK             ;[E121]
001517  030556  017010  001526          ADD     #SDI.1T,R16,BAR %JNZRO  SEEKGS  ;   THEN DO GRP SEL; BAR -> SEEK CMD       ;[E121]
001520  113740  003012  010000          MOV     #<INSEEK*256.>,BUF              ; PUT SEEK COMMAND IN HI BYTE                   ;[E121]
001521  033757  000006  121347          MOV     #SEEKL,R17      %CALL   LEV2.T  ; R17=CMD LEN/OUTPUT SEEK CMD                   ;[E121]
001522  013460  010000  062016          NOP\T                   %CNZRO  D.DNSK  ; [U52EC2][ECO#2]  ELSE DECR # OF SEEKS CNTR
001523  013451  010011  011714          TST     R11             %JNZRO  D.ECKS  ; [U52EC2][U52EC1]IF NOT SENT THEN GO CHK FOR ERRORS
                                ; ***   SEND SUCCEEDED - FORCE D.VECT BACK TO D.IDLE FOR OVERLAPPING ***
001524  013440  000000  132036          NOP                     %CALL   D.INS1  ; [ECO#2]  ELSE OK & STEP TO RECEIVE VECTOR;[E121]
001525  013440  000000  102110          NOP                     %JMP    D.TIMR  ; RESET TIMER AND RETURN                        ;[E121]
                                                                                                                                ;[E121]
                                ;       Code to issue group select when appropriate instead of a seek                           ;[E121]
                                ;       This code terminates vector processing immediately. The I/O routines                    ;[E121]
                                ;       D.READ or D.WRIT will wait for R/W Ready to come up, as group selects                   ;[E121]
```



```
                                ;       are fast and the extra latency of using C.SEEK hurts spiralling                 ;[E121]
                                ;       performance.                                                                    ;[E121]
                                                                                                                        ;[E121]
001526  010555  007013  000000  SEEKGS: ADD     #SDI.GP,DBAR\N,BAR              ; get current group number              ;[E121]
001527  033712  000003  122016          MOV     (BUF),R12       %CALL   D.DNSK  ; Always downcount NSEEKS               ;[E121]
001530  133552  000216  131212          BIS     #SGRPCD,R12     %CALL   LEVOWR  ; Send group sel to drive               ;[E121]
001531  013451  010011  011714          TST     R11             %JNZRO  D.ECKS  ; If not sent, check for errors         ;[E121]
001532  133751  000002  010000          MOV     #1000,R11                       ; set count to avoid infinite wait      ;[E121]
001533                                  ASSUME  RWRDY,EQ,BIT15                                                          ;[E121]
001533  013711  010002  101675  10$:    MOV     (RTDS),R11\N    %JZRO   CSEEKA  ; Wait for R/W Ready to drop            ;[E121]
001534  031451  030011  101533          DEC     R11             %JMSB   10$     ; So we don't start I/O too soon!       ;[E121]
001535  013440  000000  111675          NOP                     %JMP    CSEEKA  ; declare the Seek complete             ;[E121]

                                ; ***   RECEIVE SEEK RESPONSE ***
                        R.SEEK::MOV     #INPUT,R13      %JMP    R.SEEK  ; EXECUTED IN JUMP TABLE
001536  033754  000007  131273          MOV     #INPLEN,R14     %CALL   LEV1RD  ; R14=RESPONSE BUFFER LENGTH
001537  013440  000000  122016          NOP                     %CALL   D.DNSK  ; [16K]DECR 'NSEEKS'
001540  013451  000011  101704          TST     R11             %JMP    D.ECKR  ; [U52EC1][16K]TEST RESPONSE STATUS
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       C.XXXX  (CHECK FOR CERTAIN CONDITION(S) TO EXIST)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL CHECK TO SEE IF CERTAIN CONDITIONS EXIST,
                                ; I.E., DUPLICATE UNIT NUMBERS, CHECK IF AVAILABLE CAN BE PERFORMED, ETC.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;       R11             SDI.PQ
                                ;       R12             SDI.S1
                                ;       R13             SDI.CP
                                ;       R16             SDI.S2
                                ;       R17             SDI.CW
                                ;       BAR             ->SDI.CW
                                ;       DPF             RESET
                                ;
                                ; OUTPUTS:
                                ;       ROUTINE DEPENDENT
                                ;
                                ;;-

                                ; [U52EC2] REMOVED FOR BETTER ERROR HANDLING ***
                                ; ***   CHECK IF MSCP AVAILABLE COMMAND CAN BE DONE ***
                                ;C.AVAL:;BIS     #DRV.AV,R17,BUF %JMP    D.INS1  ; [U52EC2]EXECUTED IN JUMP TABLE
                                ;       ADD     #P.MOD,R11\N,BAR %CALL  S.LD14  ; R14=MODIFIERS FIELD
                                ;       MOV     #UN.BUF&LOBYT,R12               ; [16K]R12=START OF SUBUNIT CHAR BLKS
                                ;       BIS     #UN.BUF&HIBYT,R12               ; [16K]CONSTRUCT UN.BUF IN R12
                                ; ***   NOW CHECK FOR FOR SISTER SUBUNIT'S STILL ONLINE ***
                                ;CAVALA:        ADD     #SDI.SD,R12\N,BAR %JZRO D.INS1  ; IF DONE THEN EXIT/ELSE BAR=UNIT'S PARENT SDI PTR
                                ;       XOR     (BUF),DBAR\N                    ; IF NOT THE SAME AS REQUEST
                                ;       MOV     #SC.SCN,R13     %JNZRO  CAVALB  ;   THEN SKIP/R13=STILL CONNECTED CODE
                                ;       ADD     #SDI.CW,R12\N,BAR %CALL S.LD11  ; R11 = CONTROL WORD
                                ; ***   SAME SDI POINTER - CHECK IF ONLINE ***
                                ;       BIT     #DRV.AV,R11     %JZRO   CAVALB  ; IF NOT ACTIVE THEN CONTINUE
                                ;       TST     R14             %JNZRO  CAVALB  ; IF AVAILABLE THEN CONTINUE
                                ; ***   CHECK FOR SPIN DOWN MODIFIER ***
                                ;       ASSUME  MD.SPD,EQ,BIT00                 ; IF SPIN DOWN REQUESTED
                                ;       BIS\T   #SC.SDI,R13     %TLSB           ;   THEN SET SPIN DOWN IGNORED
                                ;       ADD     #SDI.SW,DBAR\N,BAR %CALL S.LD12 ; GET CODE IN STATE WORD
                                ;       ASSUME  ST.SUC,EQ,0                     ; MAKE SURE SUCCESS=0
                                ;       MOV\T   R13,BUF         %TZRO           ; IF EQ 0 THEN SET SUBCODE IN
                                ;       MOV     #XCMP,R12       %JMP    D.DONE  ; GO FINISH THIS CHAIN
                                ;CAVALB:        ADD     #SDI.LL,R12     %CALL   S.SD11  ; UPDATE R12 TO NEXT SUBUNIT BLK
                                ;       XOR     R12,R11         %JMP    CAVALA  ; CHECK AGAINST END AND LOOP

                                ; ***   PROCESS NEW UNIT GET STATUS RESPONSE AND CHECK FOR DUPLICATES ***
                                C.DUPL:;MOV     #SDI.4&LOBYT,R17 %JMP   C.DUPL  ; EXECUTED IN JUMP TABLE
001541  133557  000003  010000          BIS     #SDI.4&HIBYT,R17                ; R17=LAST SDI CONTROL BLK ADDR
001542  034456  000016  000000          CLR     R16                             ; R16=0,FLAG FOR UNIT NUMBER DEACTIVATION
                                ; ***   NOW LET'S CHECK FOR DUPLICATES ! ***
                                ; ***   DBAR POINTS TO THIS UNIT'S SDI BLOCK, R17 POINTS TO OTHERS SDI BLOCK ***
001543  010555  007043  133604  CDUPLA: ADD     #SDI.UN,DBAR\N,BAR %CALL S.LD12 ; GET THIS UNIT'S SUBUNIT/UNIT WORD IN R12
```



```
001544  114552  000014  010000          BIT     #<BIT10!BIT11>,R12              ; IF UNIT # TOO BIG
001545  013460  010400  051566          NOP\T           @SDPF   %JNZRO  CDUPLE  ;   THEN TREAT AS DUPLICATE
                                ;       AND     #DRV.SU,R12     %CALL   S.LD11  ; [U52EC3]R12=SUBUNIT MASK, R11=UNIT #
001546  010557  007043  133605          ADD     #SDI.UN,R17\N,BAR %CALL S.LD13  ; GET OTHER UNIT'S SUBUNIT/UNIT WORD IN R13
                                ; ***   FOLLOWING REMOVED FOR MULTI-UNIT REMOVAL [U52EC3] ***
                                ;       SUB     R13,R11                         ; GET DIFFERENCE BETWWEN THE UNIT NUMBERS
                                ;       ADD     #4,R11          %JZRO   CDUPLB  ; IF ZERO THEN MATCH/ELSE BIAS DIFFERENCE BY 4
                                ;       BIC     #DRV.UM,R11                     ; ISOLATE UNIT NUMBER FIELD OF R11
                                ;       BIC     #7,R11\N                        ; CHECK FOR BIASED DIFFERENCE < 4, MEANING
                                ;                                               ; UNBIASED DIFFERENCE IS IN THE RANGE [-4,+3]
                                ;       ADD     #12.,R11        %JNZRO  CDUPLD  ; BR IF UNIT NUMBERS GROSSLY DIFFERENT,
                                ;                                               ; CHANGE DIFF BIAS TO 16 (TO RANGE [12,19])
                                ;       AND     #DRV.SU,R13     %CALL   S.ROTL  ; ROTATE OTHER MASK BY BIASED DIFFERENCE VALUE
                                ;                                               ; THIS ALIGNS THE BITS OF THE TWO UNIT MASKS
                                ;                                               ; SO THAT BITS CORRESPONDING TO THE SAME
                                ;                                               ; SUBUNIT NUMBER ARE IN THE SAME BIT POSITION.
001547  036153  000012  000000          XOR     R12,R13                         ; [U52EC3]CHECK FOR DUPLICATES
001550  135553  000374  000000  CDUPLB: BIC     #DRV.UM,R13                     ; [U52EC3]CLEAR OFF NON-UNIT BITS
001551  016157  010015  011563          XOR     DBAR,R17\N      %JNZRO  CDUPLD  ; [U52EC3]IF NONE, FORGET IT / CHECK FOR SAME SDI
001552  133754  010034  101583          MOV     #<DRVOL!DRAVL!DRDUP>,R14 %JZRO CDUPLD ; IF SAME, WE WASTED OUR TIME/GET ONLINE MASK
001553  010557  007000  133567          ADD     #SDI.ST,R17\N,BAR %CALL P.LOCK  ; GET OTHER SDI'S STATUS
001554  015514  000003  000000          BIC     (BUF),R14\N                     ; FIRST CHECK IF HE IS UNKNOWN (BITS = 11)
001555  033714  010403  141562          MOV\F   (BUF),R14 @SDPF %JZRO   CDUPLC  ; IF UNKNOWN THEN IGNORE HIM
001556  114554  000030  010000          BIT     #<DRAVL!DRVOL>,R14              ; IF HE'S ONLINE
001557  033176  010017  151560          BIS\T   R17,R16         %TZRO           ;   THEN MAKE R16 NONZERO
001560  114554  000004  000000          BIT     #DRDUP,R14                      ; IF HE'S AVAILABLE SEE IF WE SHOULD
001561  133574  013005  151562          BIS\T   #<DATT!DRDUP>,R14,BUF %TZRO     ; GET ON HIS CASE WHEN WE GET AROUND TO IT
001562  133544  000200  010000  CDUPLC: BIS     #PLOCK,RLL                      ; RELEASE THE MUTEX VARIABLE
001563  013440  000000  123600  CDUPLD: NOP                     %CALL   S.SD11  ; R11=SDI.1
001564  036151  000017  010000          XOR     R17,R11                         ; SEE IF WE HAVE DONE ALL OF THE SDI CTL BLKS
001565  131557  010120  011543          SUB     #SDIB.L,R17     %JNZRO  CDUPLA  ; LOOP UNTIL WE HAVE....
                                ; ***   DONE WITH DUPLICATE CHECK ***
001566  133752  000034  133572  CDUPLE: MOV     #<DRAVL!DRVOL!DRDUP>,R12 %CALL D.GMST ; R11=MUTEXED SDI STATUS
001567  035152  000011  010000          BIC     R11,R12                         ; IF ALL BITS SET
001570  135571  010024  141571          BIC\T   #<DRAVL!DRDUP>,R11 %TZRO        ;   THEN MARK OFFLINE
001571  133571  060014  151572          BIS\T   #<DRVOL!DRDUP>,R11 %TDPF        ; IF DUPLICATE THEN MARK
001572  135571  060020  151573          BIC\T   #DRAVL,R11      %TDPF           ;   OFFLINE & DUPLICATE
001573  013440  003011  133570          MOV     R11,BUF         %CALL   UNLOCK  ; RESET STATUS/FREE U.PROC
001574  010555  007043  133604          ADD     #SDI.UN,DBAR\N,BAR %CALL S.LD12 ; R12=UNIT NUMBER OF THIS DRIVE
001575  013456  000016  000000          TST     R16                             ; IF R16 NEQ 0 (DEACTIVATE THIS UNIT NUMBER)
001576  135572  013374  051577          BIC\T   #DRV.UM,R12,BUF %TNZRO          ;   THEN DEACTIVATE UNIT NUMBER
                                                                                ; BECAUSE DUPLICATE IS ONLINE
                                ; ***   CHECK DRIVE STATUS FOR AVAILABLE PREVENTING CONDITIONS ***
001577  010555  007044  123604          ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD12 ; R12=DRIVE STATUS WORD 1
001600  010555  007045  133610          ADD     #SDI.S2,DBAR\N,BAR %CALL S.LD16 ; R16=DRIVE STATUS WORD 2
001601  114551  000010  000000          BIT     #DRVOL,R11                      ; IF DRIVE AVAILABLE
001602  014552  010200  101660          BIT     #DRV.OA,R12     %JZRO   C.SUBC  ;   THEN CHK FOR ATTN'S/IF DRIVE ALREADY ONLINE
001603  114551  010004  001624          BIT     #DRDUP,R11      %JNZRO  C.AONL  ;   THEN SEND ATTN/IF DRIVE DUPLICATE
001604  114556  010010  011627          BIT     #DRV.C1,R16     %JNZRO  C.DUPX  ;   THEN SEND ATTN/IF DRIVE SICK
001605  033714  010002  012074          MOV     (RTDS),R14      %JNZRO  D.ZATT  ;   THEN EXIT/ELSE R14=DRIVE STATE
001606  014554  000100  010000          BIT     #AVAIL,R14                      ; IF DRIVE ONLINE TO ME
                                ; ***   ALREADY ONLINE TO ME - MUST DISCONNECT ***
001607                                  ASSUME  V.DCON,EQ,V.STAT+1
001607  033751  010004  112036          MOV     #<V.GCHR-V.STAT>,R11 %JZRO D.INS1;  THEN DISCONNECT
```

```
KDBDP           DIGITAL EQUIPMENT CORP., 2901 ASSEMBLER VERSION 32              PAGE  154
SDI LEVEL 0,1, AND 2 ROUTINES

                                ; ***   DRIVE POTENTIALLY AVAILABLE ***
001610  010555  007043  123606            ADD     #SDI.UN,DBAR\N,BAR %CALL S.LD14 ; R14=UNIT NUMBER
001611  010555  007001  000000            ADD     #SDI.SL,DBAR\N,BAR              ; BAR=PTR TO SDI.SL
001612  135554  003354  112037            BIC     #<DRV.UM-DRV.U1>,R14,BUF %JMP D.INSV ; SAVE 10 BITS OF UNIT NUMBER IN SDI.SL

                                ; ***   CHECK IGNORE MEDIA FORMAT FLAG
001613  010555  007022  133605  C.IMFF: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD13 ; [V05] R13 -> MSCP PACKET
001614  030553  007007  133605            ADD     #P.MOD,R13,BAR  %CALL   S.LD13  ; [V05] R13 -> MODIFIER FIELD/STORE MOD FIELD IN R1
001615  014553  010002  112036            BIT     #MD.IMF,R13     %JZRO   D.INS1  ; [V05] IF ZERO,NEXT STEP/CHECK BIT
001616  033753  010002  152036            MOV\F   #SEEK,R13       %JZRO   D.INS1  ; [V05] IF CLEAR,INC TO NEXT STEP
001617  133553  000002  010000            BIS     #GSEL,R13                       ; clear special seek flag too           ;[E121]
001620  010555  007007  133604            ADD     #SDI.DB,DBAR\N,BAR %CALL S.LD12 ; [V05] R12 -> BUF CNTR BLK
001621  033452  007012  133604            MOV     R12,R12,BAR     %CALL   S.LD12  ; [V05] R12 = (BUF CNTR BLK)
001622  133552  003100  132011            BIS     #BFULL,R12,BUF  %CALL   D.CLRS  ; [V05] SET BUFFER FULL/CLEAR SEEK BIT IN STATUS
001623  033752  000100  102022            MOV     #BFSV,R12       %JMP    D.DONE  ; [V05] GO TO DONE/SET BFSV

                                ; ***   ACCESS PATH PROCESSING ***
001624  033752  010101  011630  C.AONL: MOV     #OP.DUP,R12     %JNZRO  C.ATTN  ; IF DUPLICATE THEN R12=DUPL OP CODE/CONTINUE
001625  013440  000000  122084            NOP                     %CALL   D.OFFL  ;   ELSE MARK DRIVE OFFLINE
001626  033752  000102  111630            MOV     #OP.ACP,R12     %JMP    C.ATTN  ; AND R12=ACCESS PATH OP CODE

                                ; ***   DUPLICATE UNIT NUMBER PROCESSING ***
001627  033752  000101  010000  C.DUPX: MOV     #OP.DUP,R12                     ; [V05]
001630                                  ASSUME  C.ATTN,EQ,.                     ; MAKE SURE C.ATTN IS NEXT

                                ; ***   SET SLAT FOR ATTENTION - U.PROC WILL CLEAR IF NOT ENABLED ***
001630  133753  000040  122013  C.ATTN: MOV     #SLAT,R13       %CALL   D.SSET  ; GO SET SLAT
001631  010555  007010  133615            ADD     #SDI.1T,DBAR\N,BAR %CALL S.ST12 ; SAVE OP CODE IN ATTN OP CODE AREA
001632                                  ASSUME  <V.DCON-V.STAT>,EQ,1            ; MAKE SURE D.INS1 IS VALID
001632  013440  000000  102036            NOP                     %JMP    D.INS1  ; INCR TO NEXT STEP

                                ; ***   DBAR = SDI CONTROL BLOCK POINTER, R13 = LOG FORMAT, R16 = MSCP STATUS, R17 --> BUFFER ***
001633  010555  007022  133603  C.LOGS: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD11 ; [U52EC2][ECO#2]R11=MSCP PKT POINTER
001634                                  ASSUME  <PKTO20&BITOO>,EQ,0             ; [16K]MAKE SURE CHECK IS VALID
001634                                  ASSUME  <<SDI.1!SDI.2!SDI.3!SDI.4>&BITOO>,NE,0; [16K]DITTO
001634                                  ASSUME  <SDI.AT&BITOO>,EQ,0             ; [16K]DITTO
001634  033751  040100  001636            MOV     #CF.MSC,R11     %JLSB   C.LOGA  ; [U52EC3]IF INTERNAL PKT THEN CHECK MISC FLAG
001635  033751  000020  000000            MOV     #CF.THS,R11                     ;   ELSE CHECK THIS HOST FLAG
001636  013740  007170  010000  C.LOGA: MOV     #CNTFLG,BAR                     ; BAR=PTR TO CONTROLLER FLAGS
001637  014511  000003  010000            BIT     (BUF),R11                       ; IF LOG NOT ENABLED
001640  133553  010100  127777            BIS     #LF.CON,R13     %RZRO           ;   THEN RETURN
001641  013440  000000  123572            NOP                     %CALL   D.GMST  ;   ELSE R11=MUTEXED SDI STATUS
001642  114551  000040  000000            BIT     #SLAT,R11                       ; IF LOG PENDING
001643  133551  013040  053570            BIS\F   #SLAT,R11,BUF   %JNZRO  UNLOCK  ;   THEN DROP THIS ONE/FREE U.PROC
001644  133544  000200  010000            BIS     #PLOCK,RLL                      ; [ECO#4][ECO#2]FREE U.PROC
001645  033451  000013  112043            MOV     R13,R11         %JMP    D.LOGF  ; [ECO#]SAVE LOG STATUS AND LOG FORMAT

                                ; ***   CHECK IF SUBUNIT PRESENT ROUTINE ***
                        C.SUBU:;ADD     #SDI.UN,DBAR\N,BAR %JMP C.SUBU  ; EXECUTED IN JUMP TABLE
001646  033714  000003  123572            MOV     (BUF),R14       %CALL   D.GMST  ; R11=MUTEXED SDI STATUS
                                                                                ; R12=STATUS SUBUNIT/UNIT WORD
001647  135551  000110  010000            BIC     #<ERRIP!DRVOL>,R11              ; CLEAR DRIVE OFFLINE
001650  133551  003020  133570            BIS     #DRAVL,R11,BUF  %CALL   UNLOCK  ; MARK DRIVE AVAIL/FREE U.PROC
001651  010555  007012  123625            ADD     #SDI.CH,DBAR\N,BAR %CALL INITM1 ; SET HI CYL SO SEEK WILL OCCUR

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```

```
KDBDP           DIGITAL EQUIPMENT CORP., 2901 ASSEMBLER VERSION 32              PAGE  155
SDI LEVEL 0,1, AND 2 ROUTINES

001652  134557  000360  122023            AND     #DRV.SU,R17     %CALL   D.GUNF  ; R17=SUBUNIT MASK/GO SET UNIT FLAGS
001653  010555  007001  123605            ADD     #SDI.SL,DBAR\N,BAR %CALL S.LD13 ; R13=SUBUNIT MASK/UNIT NUMBER
001654  174553  000160  123604            AND\L   #<DRV.SU-DRV.U4>,R13 %CALL S.LD12; R13=MASK/R12=UNIT NUMBER
001655  054153  000014  010000            AND\R   R14,R13                         ; CHECK IF UNIT PRESENT
001656  130172  010013  063615            ADDC\T  R13,R12         %CNZRO  S.ST12  ; SET IN NEXT SUBUNIT CODE, INCREMENTING UNIT #
001657  131751  010003  002037            NEG     #<V.GCR2-V.GCR1>,R11 %JNZRO D.INSV ; ADJUST STATE VECTOR OFFSET TO LOOP
                                ; ***   DONE - SET SLAT FOR AVAILABLE ATTENTION MESSAGE ***
001660  010555  007045  133610  C.SUBC: ADD     #SDI.S2,DBAR\N,BAR %CALL S.LD16 ; R16=2ND WORD DRIVE STATUS
001661  010555  007044  133603            ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD11 ; R11=1ST WORD DRIVE STATUS
001662                                  ASSUME  DRV.RU,EQ,BITOO                 ; MAKE SURE DRV.RU IS LSB
001662  114556  040360  112036            BIT     #DRV.SN,R16     %JNLSB  D.INS1  ; IF NOT SPINNING THEN EXIT
001663  033752  010100  002036            MOV     #OP.AVA,R12     %JNZRO  D.INS1  ; IF S BIT SET THEN NO ATTN MSG
001664  013440  000000  101630            NOP                     %JMP    C.ATTN  ;   ELSE GO DO ATTENTION MSG

                                ; ***   CHECK FOR NEED TO SEND DISCONNECT COMMAND (ATTENTION) ***
                        C.DCON:;COM     R12,R17         %JMP    C.DCON  ; EXECUTED IN JUMP TABLE
001665  014557  000003  000000            BIT     #<DRV.PS!DRV.RU>,R17            ; IF PORT/RUN-STOP SWITCH OUT
001666                                  ASSUME  DRV.RU,EQ,DCN.ST                ; MAKE SURE BITS COMPATIBLE
001666  034577  010001  051411            AND\T   #DRV.RU,R17     %JNZRO  S.DCN3  ;   THEN DISCONNECT
                                ; ***   CHECK GET STATUS RESPONSE FOR DRIVE ERROR ***
001667  010555  007026  123603            ADD     #SDI.ES,DBAR\N,BAR %CALL S.LD11 ; R11=EXTENDED STATUS
001670  014556  000200  000000            BIT     #DRV.DE,R16                     ; IF DRIVE ERROR SET
001671  033571  013004  051672            BIS\T   #FSEEK,R11,BUF  %TNZRO          ;   THEN FORCE SEEK NEXT I/O
001672                                  ASSUME  D.INS2,EQ,.                     ; MAKE SURE D.INS2 IS NEXT
001672  033751  000002  112037  D.INS2: MOV     #2,R11          %JMP    D.INSV  ; R11=2 (INCREMENT VALUE)

                                ; ***   SEEK COMPLETE CHECK ***
                        C.SEEK:;MOV     #2,R13          %JMP    C.SEEK  ; EXECUTED IN JUMP TABLE
001673  033711  140002  111715            MOV     (RTDS),R11      %JDSER  D.ERCK  ; IF ERROR THEN RECOVER/ELSE R11=DRIVE STATUS
001674                                  ASSUME  RWRDY,EQ,BIT15                  ; MAKE SURE RWRDY IS MSB
001674  013440  030000  011715            NOP                     %JNMSB  D.ERCK  ; [U52EC2]IF NO R/W RDY THEN CHK TIMEOUT
                                                                                ; [U52EC2]  ELSE SEEK OPERATION COMPLETE
                                ; ***   SEEK OPERATION COMPLETE ***
001675  010555  007020  123603  CSEEKA: ADD     #SDI.EO,DBAR\N,BAR %CALL S.LD11 ; R11=LEVEL 0 ERROR STATE
001676                                  ASSUME  ERRINP,EQ,BIT15                 ; MAKE SURE ERRINP IS MSB
001676  114551  033077  112005            AND     #<IORTY+LV2CNT>,R11\N,BUF %JMSB D.INSO  ; [ECO#2]IF ACTIVE THEN CONTINUE (ADD 0 TO STATE VEC
001677  033752  000001  000000            MOV     #PKIP,R12                       ; [U52EC2]R12=BIT TO SET
001700  010555  007022  133603            ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=PKT PTR
001701                                  ASSUME  <PKTO20&BITOO>,EQ,0             ; [U52EC2]MAKE SURE CHECK IS VALID
001701                                  ASSUME  <<SDI.1!SDI.2!SDI.3!SDI.4>&BITOO>,NE,0; [U52EC2]DITTO
001701                                  ASSUME  <SDI.AT&BITOO>,EQ,0             ; [U52EC2]DITTO
001701  010555  047020  033623            ADD     #SDI.EO,DBAR\N,BAR %CLSB S.CLRB ; [U52EC2]IF ATTN THEN CLR SDI.EO
001702  033753  000042  000000            MOV     #<SEEK!SUSP>,R13                                                ;[E121]
001703  133553  000002  102015            BIS     #GSEL,R13       %JMP    D.DONZ  ; R13=BITS TO CLEAR, INCLUDING GSEL     ;[E121]

                                .PAGE

KDBDP           KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```

LSCS FORM=QUAD

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       D.ECKR  (ERROR CHECK FOR RECEIVE SDI LEVEL TWO RESPONSE)
                                ;       D.ECKS  (ERROR CHECK FOR SEND SDI LEVEL TWO COMMAND)
                                ;       D.ERRO  (LEVEL 0 ERROR RECOVERY PROCESS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES PERFORM ERROR RECOVERY AS SPECIFIED IN THE SDI
                                ; SPECIFICATION, INCLUDING SEND AND RECEIVE ERROR RETRY AND RECOVERY.
                                ;
                                ; INPUTS:
                                ;       ROUTINE DEPENDENT
                                ;
                                ; OUTPUTS:
                                ;       ROUTINE DEPENDENT
                                ;
                                ;;-

                                ; ***   CHECK FOR SDI RECEIVE ERRORS ***
001704  013451  010011  101711  D.ECKR: TST     R11             %JZRO   D.ECRA  ; [EC0#2][EC0#1]IF EQ 0 THEN CONT
001705                                  ASSUME  DCERR,EQ,1                      ; [EC0#1]MAKE SURE DCERR IS LSB
001705  033753  040001  101723          MOV     #1,R13          %JNLSB  D.ECKB  ; [EC0#1]IF EQ 0 THEN CONT/R13=1
                                ; ***   RECEIVE TIMEOUT - CHECK IF DRIVE HAS RECEIVER READY UP ***
001706  013711  110002  011715          MOV     (RTDS),R11\N    %JNCSR  D.ERCK  ; [EC0#2][EC0#1]TEST REAL TIME DRIVE STATE
001707                                  ASSUME  DRDY,EQ,BIT00                   ; MAKE SURE DRDY IS LSB
001707  013440  040000  111715          NOP                     %JNLSB  D.ERCK  ; IF DRV OK THEN GO CHK DRV CLKS/TIMER
001710  033751  000001  101723          MOV     #DCERR,R11      %JMP    D.ECKB  ;   ELSE SHORT CIRCUIT TIMEOUT & INIT DRV
001711  016552  000176  010000  D.ECRA: XOR     #COMPLT,R12\N                   ; CHECK FOR COMPLETED OP CODE
001712  033751  010040  142036  D.ECRB: MOV\F   #UNERR,R11      %JZRO   D.INS1  ; [U52EC2][U52EC1]IF OK THEN INCR STATE VECTOR
001713  033753  000001  111723          MOV     #1,R13          %JMP    D.ECKB  ;   ELSE GO CHK ERROR STATE VECTOR
                                ; ***   CHECK FOR SDI SEND ERRORS ***
001714  034453  010013  112036  D.ECKS: CLR     R13             %JZRO   D.INS1  ; IF EQ 0 THEN INCR STATE VECTOR
                                ; ***   CHECK FOR NO DRIVE CLOCKS ***
001715  010555  007022  123604  D.ERCK: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD12 ; R12=MSCP PKT PTR
001716  030552  007006  133612          ADD     #P.OPCD,R12,BAR %CALL   S.LL12  ; [EC0#2]R12=THIS OP CODE
001717  036552  110013  101721          XOR     #OP.DAP,R12     %JCSR   D.ECKA  ; IF CLKS THEN CONTINUE
001720  033772  015001  051766          MOV\T   #CONT,R12,RTCS  %JNZRO  D.NCLK  ; [EC0#2]  ELSE IF NOT DET ACC PATHS THEN FATAL ERROR
                                ; ***   CHECK SDI TIMER FOR EXPIRATION ***
001721  010555  007004  133604  D.ECKA: ADD     #SDI.TM,DBAR\N,BAR %CALL S.LD12 ; R12=SDI TIMER VALUE
001722  033751  010001  027777          MOV     #DCERR,R11      %RNZRO          ; IF NO TIMEOUT THEN RETURN
                                ; ***   SDI TIMER EXPIRED - CHECK IF RECOVERY IN PROGRESS ***
001723  010555  007020  133604  D.ECKB: ADD     #SDI.E0,DBAR\N,BAR %CALL S.LD12 ; R12=ERROR STATE VECTOR
001724                                  ASSUME  ERRINP,EQ,BIT15                 ; IF LEV 0 ERROR IN PROGRESS
001724  130552  030377  101730          ADDC    #377,R12        %JMSB   D.ECKC  ;   THEN INCR RETRY COUNT/CONTINUE
                                ; ***   TURN ON ERROR RECOVERY ***
001725  133552  000200  000000          BIS     #ERRINP,R12                     ; [EC0#2]TURN ON ERROR RECOVERY FLAG
                                ; ***   BACK UP ORIGINAL STATE VECTOR BY VALUE IN R13 ***
001726  010555  007021  133606          ADD     #SDI.SV,DBAR\N,BAR %CALL S.LD14 ; R14=STATE VECTOR
001727  131154  000013  123617          SUB     R13,R14         %CALL   S.ST14  ; BACK UP CORRECT AMOUNT (R13) AND RESET STATE VECT
                                ; ***   SELECT CORRECT LEVEL 0 RECOVERY VECTOR ***
001730  035552  000377  000000  D.ECKC: BIC     #LOBYT,R12                      ; ZAP ERROR RECOVERY VECTOR
001731  033552  000254  000000          BIS     #V.INIT&LOBYT,R12               ; SET IN ERROR RECOVERY VECTOR + INIT
001732  013451  000011  000000          TST     R11                             ; TEST ERROR CODE
001733                                  ASSUME  DCERR,EQ,BIT00                  ; INSURE LSB CHECK
```



```
001733  130472  040012  141735          INC\T   R12             %JNLSB  D.ECKD  ; IF NOT TIMEOUT THEN SKIP INIT
001734  133552  000100  000000          BIS     #ERRINI,R12                     ;   SET INIT DONE BIT
                                ; ***   RESET ERROR RECOVERY WORD ***
001735  010555  007020  133615  D.ECKD: ADD     #SDI.E0,DBAR\N,BAR %CALL S.ST12 ; SAVE CURRENT RECOVERY STATE
001736  134552  000003  000000          AND     #LV2CNT,R12                     ; ISOLATE RETRY COUNT
                                ; ***   SDI ERROR - R11 HAS LEVOWR/LEVORD STATUS, R12 HAS RETRY COUNT ***
001737  014551  000040  010000          BIT     #UNERR,R11                      ; IF R11 EQ UNSUCCESSFUL RESPONSE
001740  033776  010353  051744          MOV\T   #<ST.DRV+SC.DDE>,R16 %JNZRO DSDIEA;  THEN UNSUCCESSFUL RESPONSE
001741  014551  000001  010000          BIT     #DCERR,R11                      ; IF SDI TIMEOUT
001742  033776  010053  051744  D.SDIE: MOV\T   #<ST.DRV+SC.STO>,R16 %JNZRO DSDIEA; THEN SET ERROR CODE/CONTINUE
001743  033756  000113  000000          MOV     #<ST.DRV+SC.INV>,R16            ;   ELSE SET INVALID ERROR CODE
                                ; ***   CHECK FOR ERROR LOGS ENABLED ***
001744  010555  007025  123605  DSDIEA: ADD     #SDI.OM,DBAR\N,BAR %CALL S.LD13 ; [U52EC1]IF PKT BEING OVERLAPPED
001745  033754  010100  011744          MOV     #AVAIL,R14      %JNZRO  DSDIEA  ; [U52EC2][U52EC1]  THEN WAIT UNTIL DONE/R14=AVAIL BIT
                                ; ***   CHECK IF ERROR ON AVAILABLE PHASE OF FAILED ONLINE COMMAND ***
001746  010555  007022  133605          ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD13 ; [U52EC3]R13=MSCP PKT PTR
001747                                  ASSUME  <PKT020&BIT00>,EQ,0             ; [U52EC3]MAKE SURE CHECK FOR
001747                                  ASSUME  <<SDI.1!SDI.2!SDI.3!SDI.4>&BIT00>,NE,0; [U52EC3]INTERNAL PACKET
001747                                  ASSUME  <SDI.AT&BIT00>,EQ,0             ; [U52EC3]IS VALID
001747  030553  047001  001753          ADD     #P.VCID,R13,BAR %JLSB   DSDIEC  ; [U52EC3]IF INTERNAL PKT THEN SKIP CHK
001750  033713  000003  010000          MOV     (BUF),R13                       ; [U52EC3]R13=P.VCID & FLAGS
001751  014553  000002  000000          BIT     #PONER,R13                      ; [U52EC3]IF AVAIL PART OF FAILED ONLINE CMD
001752  133772  010003  051753          MOV\T   #SDIRTY*256.,R12 %TNZRO         ; [U52EC3]  THEN ERROR IS FATAL
001753  033753  000003  131633  DSDIEC: MOV     #FM.SDI,R13     %CALL   C.LOGS  ; [U52EC3]GO CHECK FOR ERROR LOGS ENABLED
                                ; ***   CHECK FOR FATAL ERROR ***
001754  136552  000003  010000          XOR     #SDIRTY*256.,R12                ; [EC0#2][EC0#1]IF NEQ TO MAX
001755  033751  010003  002112          MOV     #FM.SDI,R11     %JNZRO  D.TIME  ;   THEN CONTINUE
                                ; ***   FATAL ERROR CLEANUP PROCESSING ***
001756  133753  000024  132043          MOV     #<DRAVL!DRDUP>,R13 %CALL D.LOGF ; [U52EC2][EC0#2]R13=SDI STATUS BITS TO CLEAR
                                ;       BIS     #RVCT,R13       %CALL   D.DNSK  ; [U52EC2][U52EC1][EC0#2]DECR NSEEKS & ZAP REVECT FLAG
001757  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=SDI STATUS
001760  114551  000030  010000          BIT     #<DRAVL!DRVOL>,R11              ; [U52EC2]IF NOT ONLINE
001761  014514  010002  011763          BIT     (RTDS),R14      %JNZRO  DSDIEB  ; [U52EC2]  THEN CONTINUE/IF DRV RTDS="AVAIL"
001762  135573  010024  051763          BIC\T   #<DRAVL!DRDUP>,R13 %TNZRO       ; [U52EC2]  THEN MARK OFFLINE & UNKNOWN
001763  033553  000102  132045  DSDIEB: BIS     #<BFSV!SEEK>,R13 %CALL D.OFFD   ; [U52EC2]MARK DRIVE OFFLINE & CHK FOR DUPLICATE
001764  133553  000002  010000          BIS     #GSEL,R13                       ; clear special seek flag too           ;[E121]
001765  033752  000244  112015          MOV     #<XCMP!SUSP!DERR>,R12 %JMP D.DONZ ; SET XCMP, DERR & SUSP INTO R12/TERMINATE COMMAND

                                ; ***   NO CLOCKS FINAL FILTER - LAST CHANCE BEFORE DROPPING DRIVE ***
                                ; ***   WAIT ABOUT 10 MSECS BEFORE GIVING UP ***
001766  030552  110002  101721  D.NCLK: ADD     #2,R12          %JCSR   D.ECKA  ; [EC0#2]IF CLKS THEN CONTINUE
001767  013440  030000  001766          NOP                     %JNMSB  D.NCLK  ; [EC0#2]  ELSE IF NEQ MSB THEN LOOP
001770                                  ASSUME  D.SDIF,EQ,,                     ; [EC0#2]MAKE SURE D.SDIF IS NEXT
                                ; ***   FATAL SDI ERROR EXIT ***
001770  133752  000003  111742  D.SDIF: MOV     #SDIRTY*256.,R12 %JMP   D.SDIE  ; [EC0#2][EC0#1]MARK OFFLINE/FINISH UP

                                ; ***   FATAL ERROR FROM S.SEEK.                                 [E126]
                                ; ***   We must decrement NSEEK before continuing or face a dead lock situation [E126]
001771  013440  000000  122016  D.SDIG: NOP                     %CALL   D.DNSK  ; DECREMENT NSEEK       [E126]
001772  013440  000000  101770          NOP                     %JMP    D.SDIF  ; GO REPORT ERROR       [E126]

                                ; ***   LEVEL 0 AND I/O ERROR RECOVERY ROUTINE ***
                                D.ERRO:;MOV     #<V.ATT2-V.ERRC>,R11 %JMP D.ERRO; EXECUTED IN JUMP TABLE
001773  010555  007020  123605          ADD     #SDI.E0,DBAR\N,BAR %CALL S.LD13 ; [EC0#2]R13=LEVEL 0 STATE VECTOR
```

```
001774                                  ASSUME  ERRINP,EQ,BIT15                 ; MAKE SURE ERRINP IS MSB
001774  114553  033077  012006          AND     #<IORTY+LV2CNT>,R13\N,BUF %JNMSB D.EROA ; [ECO#2]CLEAR ALL BUT I/O & SDI RETRY
001775  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS
001776  014551  000004  010000          BIT     #DERR,R11                       ; IF HARD ERROR
001777  114551  010030  041770          BIT\F   #<DRVOL+DRAVL>,R11 %JNZRO D.SDIF;   THEN MARK OFFLINE/CLEAN UP
002000  014552  010020  012005          BIT     #DRV.SR,R12     %JNZRO  D.INSO  ; [ECO#2]IF DRV NOT ONLINE THEN DONE
                                                                                ; [ECO#2]IF SPINDLE NOT AT SPEED THEN DONE
                                ; ***   DRIVE ONLINE - CHECK IF ERROR WAS DRIVE ERROR ***
002001  014556  010200  112005          BIT     #DRV.DE,R16     %JZRO   D.INSO  ; IF NOT DRIVE ERROR (R16 LOADED IN D.VECP)
002002  114573  010100  142003          BIT\T   #ERRINI,R13     %TZRO           ;   [ECO#2]THEN CHECK IF INIT DONE
002003  030553  010003  102005          ADD     #<V.ERSK-V.ERRC>,R13 %JZRO D.INSO ; [ECO#2]THEN SKIP PAST RECALIBRATE
002004  010555  007020  133616          ADD     #SDI.EO,DBAR\N,BAR %CALL S.ST13 ;   [ECO#2]ELSE RESTORE LEVEL 0 VECTOR
002005  034451  000011  112037  D.INSO: CLR     R11             %JMP    D.INSV  ; AND DO RE-SEEK
                                ; ***   I/O ERROR RECOVERY PROCESSING ***
002006  114553  010010  112037  D.EROA: BIT     #IOSEK,R13      %JZRO   D.INSV  ; [ECO#2]IF ATTN THEN CONNECT TO ATTENTION CHAIN
002007  013440  010000  002036          NOP                     %JNZRO  D.INS1  ; IF SEEK RECOVERY THEN DO RECAL
002010  033751  000003  102037          MOV     #<V.ERSK-V.ERRC>,R11 %JMP D.INSV; IF OTHER RECOVERY THEN JUST DO SEEK
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       D.XXXX  (PERFORM UTILITY FUNCTION FOR VECTOR PROCESSES)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL PERFORM UTILITY FUNCTIONS FOR THE VECTOR
                                ; STATE PROCESSES; SUCH AS INCREMENTING STATE VECTORS, OBTAINING BUFFERS
                                ; FROM SUBUNIT CHARACTERISTICS BUFFER POOL, RELEASING BUFFERS TO THAT SAME
                                ; POOL, MARKING A DRIVE OFFLINE, ETC.
                                ;
                                ; INPUTS:
                                ;       ROUTINE DEPENDENT
                                ;
                                ; OUTPUTS:
                                ;       ROUTINE DEPENDENT
                                ;
                                ;;-

                                ; ***   ROUTINE TO CLEAR BITS (PASSED IN R13) IN SDI STATUS - MUTEXED ***
002011  013440  000000  123572  D.CLRS: NOP                     %CALL   D.GMST  ; R11=MUTEXED SDI STATUS
002012  035151  003013  103570          BIC     R13,R11,BUF     %JMP    UNLOCK  ; ZAP DATT/RESET/FREE U.PROC

                                ; ***   ROUTINE TO SET BITS (PASSED IN R13) IN SDI STATUS - MUTEXED ***
002013  013440  000000  123572  D.SSET: NOP                     %CALL   D.GMST  ; R11=MUTEXED SDI STATUS
002014  033151  003013  103570          BIS     R13,R11,BUF     %JMP    UNLOCK  ; SET BIT IN R13/RELEASE U.PROC/RETURN

                                S.CL11: ; ***   SUBROUTINE TO CLEAR R11 ***
002015  034451  000011  127777  D.DONZ: CLR     R11             %RET            ; ZAP R11/RETURN


                                ; ***   NUMBER OF SEEKS WORD (NSEEKS) DECREMENT ROUTINE ***
002016  013740  007156  133567  D.DNSK: MOV     #NSEEKS,BAR     %CALL   P.LOCK  ; LOCK OUT U.PROC
002017  033714  000003  000000          MOV     (BUF),R14                       ; R14=NUMBER OF SEEKS TO START
002020  031474  010014  063617          DEC\T   R14             %CNZRO  S.ST14  ; IF NEQ 0 THEN DECREMENT AND RESET
002021  133544  000200  137777          BIS     #PLOCK,RLL      %RET            ; UNLOCK U.PROC/RETURN

                                ; ***   STATE VECTOR COMPLETION ROUTINE ***
                                D.DONE:;MOV     #<SDI.ST FLAG>,R12 %JMP D.DONE  ; EXECUTED IN JUMP TABLE
002022  034453  000013  127777          CLR     R13             %RET            ; ZAP R13/RETURN

                                ; ***   GET UNIT FLAGS ROUTINE (R17=SUBUNIT MASK) ***
002023  010555  007067  123603  D.GUNF: ADD     #SDI.RT,DBAR\N,BAR %CALL S.LD11 ; R11=REMOVABLE MEDIA FLAG
002024  034551  000200  000000          AND     #UNC.RM,R11                     ; ISOLATE FLAG
002025  010555  007060  123610          ADD     #SDI.UF,DBAR\N,BAR %CALL S.LD16 ;[V05] R16=UNIT FLAGS THIS UNIT
002026  135556  000040  010000          BIC     #UF.WPH,R16                     ; ZAP H/W WRITE PROTECT FLAG
002027  014152  000017  010000          BIT     R17,R12                         ; IF WRITE PROTECT FOR THIS UNIT
002030  133576  010040  042031          BIS\T   #UF.WPH,R16     %TNZRO          ;   THEN SET WRITE PROT UNIT FLAG
002031  033156  003011  127777          OR      R11,R16,BUF     %RET            ; OR IN REMOVABLE FLAG/RESET FLAGS/RETURN

                                ; ***   GET SUBUNIT CHARACTERISTICS BUFFER FROM FREE POOL ***
                                D.GSCB:;MOV     #UN.BUF&LOBYT,R12 %JMP  D.GSCB  ; [16K]EXECUTED IN JUMP TABLE
                                ; ***   EMPTY BUFFER FOUND - SET IN PARAMETERS ***
002032  010555  007001  123603          ADD     #SDI.SL,DBAR\N,BAR %CALL S.LD11 ; R11=SUBUNIT CODE/UNIT NUMBER
002033  133551  000014  010000          BIS     #<DRV.AV+DRV.AT>,R11            ; INDICATE DRIVE AVAILABLE AND NEEDING ATTENTION
```

```
002034  010555  007056  133614          ADD     #SDI.CW,DBAR\N,BAR %CALL S.ST11 ; SAVE IN BUFFER CONTROL WORD
002035  010555  007057  133620          ADD     #SDI.SD,DBAR\N,BAR %CALL S.STDB ; SAVE SDI CNTRL BLK PTR
002036                                  ASSUME  D.INS1,EQ,.                     ; *** NOTE - FALL INTO D.INS1

                        ; ***   INCREMENT STATE VECTOR ROUTINE ***
002036  033751  000001  112037  D.INS1: MOV     #1,R11          %JMP    D.INSV  ; R11=1 (INCREMENT VALUE)
002037  010555  007020  133604  D.INSV: ADD     #SDI.EO,DBAR\N,BAR %CALL S.LD12 ; R12=LEVEL 0 ERROR STATE VECTOR
002040                                  ASSUME  ERRINP,EQ,BIT15
002040  010575  037021  073604          ADD\T   #SDI.SV,DBAR\N,BAR %CNMSB S.LD12; IF ERROR RECOV NOT ACTIVE THEN R12=STATE VECTOR
002041  030152  000011  123615          ADD     R11,R12         %CALL   S.ST12  ; BUMP TO NEXT STATE
002042  034452  000012  102022          CLR     R12             %JMP    D.DONE  ; ZAP R12/GO TO DONE

                        ; ***   CHANGE TO FATAL ERROR LOG, R11=LOG FORMAT,R16=MSCP STATUS ***
002043  010555  007010  123614  D.LOGF: ADD     #SDI.1T,DBAR\N,BAR %CALL S.ST11 ; SAVE ERROR LOG FORMAT
002044  010555  007003  103621          ADD     #SDI.SW,DBAR\N,BAR %JMP S.ST16  ; SAVE FATAL ERROR STATUS

                        ; ***   ZAP UNIT NUMBER, CHECK FOR DUPLICATES, MARK DRIVE OFFLINE ***
002045  010555  007043  000000  D.OFFD: ADD     #SDI.UN,DBAR\N,BAR              ; ZAP UNIT NUMBER to 4095[rae06]
002046  117740  003360  000000          COM     #170000,BUF                     ;                           [rae06]
002047  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS WORD
002050  114551  000004  000000          BIT     #DRDUP,R11                      ; IF DRIVE WASN'T DUPLICATE
002051  013440  010000  102064          NOP                     %JZRO   D.OFFL  ;   THEN CONTINUE
002052  033754  000004  123600          MOV     #4,R14          %CALL   S.SD11  ; R11=1ST SDI PTR
002053  010551  007000  133567  DOFFDA: ADD     #SDI.ST,R11\N,BAR %CALL P.LOCK  ; BAR=SDI STATUS PTR/LOCK OUT U.PROC
002054  033712  000003  000000          MOV     (BUF),R12                       ; R12=SDI STATUS
002055  114552  000004  000000          BIT     #DRDUP,R12                      ; IF THIS ONE NOT A DUPLICATE
002056  114552  010020  102061          BIT     #DRAVL,R12      %JZRO   DOFFDB  ;   THEN CONTINUE
002057  135552  010004  042061          BIC\F   #DRDUP,R12      %JNZRO  DOFFDB  ; IF DRAVL SET THEN CONTINUE
002060  133552  003001  000000          BIS     #DATT,R12,BUF                   ;   ELSE SET DATT IN SDI STATUS
002061  133544  000200  010000  DOFFDB: BIS     #PLOCK,RLL                      ; UNLOCK U.PROC
002062  031454  000014  000000          DEC     R14,R14                         ; DECR R14
002063  030551  010120  002053          ADD     #SDIB.L,R11     %JNZRO  DOFFDA  ; IF R14 NEQ 0 THEN BUMP R11/LOOP

                        ; ***   RELEASE SUBUNIT BUFFERS AND MARK DRIVE OFFLINE OR UNKNOWN ***
002064  010555  007060  123623  D.OFFL: ADD     #SDI.UF,DBAR\N,BAR %CALL S.CLRB ;[mjt016][UQA]CLEAR UNIT FLAGS WORD
002065  133752  000034  133572          MOV     #<DRAVL!DRVOL!DRDUP>,R12 %CALL D.GMST ; R11=MUTEXED SDI STATUS
002066  014551  000004  010000          BIT     #DERR,R11                       ; IF DRIVE ERROR
002067  135572  010024  042070          BIC\T   #<DRAVL!DRDUP>,R12 %TNZRO       ;   THEN CLEAR DRAVL AND DRDUP
002070  033151  003012  113570          BIS     R12,R11,BUF     %JMP    UNLOCK  ; RESET STATUS/FREE U.PROC/RETURN
                        ; ***  RELEASE SUBUNIT CHARACTERISTICS BUFFERS FOR SDI CONTROL BLOCK IN DBAR ***

                        ;;;D.RELB:      ADD     #SDI.CW,DBAR\N,BAR %CALL S.LD11 ; [UQA]R11=CONTROL WORD
                        ;;;             BIC     #DRV.UM,R11,BUF                 ; [UQA]FREE UP SUBUNIT CHAR BLK
                        ;;;             ADD     #SDI.UF,DBAR\N,BAR %JMP S.CLRB  ; [UQA]CLEAR UNIT FLAGS WORD

                        ; ***   DO READ FOR ONLINE COMMAND ***
                        ; ***   READ FORMAT CONTROL TABLE FOR FORMAT INFORMATION ***
                        D.OLRD:;NOP                     %JMP    D.OLRD  ; EXECUTED IN JUMP TABLE
002071  033753  000050  132011          MOV     #<VECT+SUSP>,R13 %CALL  D.CLRS  ; R13=BITS TO CLR/GO CLR SDI STATUS
                        ; ***   D.READ DOES NOT RETURN BUT RATHER JUMPS BACK TO D.IDLE, THIS LEAVES
                        ; ***   A STACK LEVEL UNPOPPED BUT SINCE THE 2901 STACK WRAPS AROUND IT SHOULD
                        ; ***   NOT CAUSE ANY PROBLEMS
002072  013440  000000  102600          NOP                     %JMP    D.READ  ; ZAP VECTOR AND SUSPEND FLAGS/GO READ SERIAL #
```



```
                        ; ***   WAIT ABOUT 20 MSECS FOR DRIVE STATE TO CHANGE ***
                        D.WAIT:;XNOR    R11,R11                         ; EXECUTED IN JUMP TABLE
002073  031451  010011  012073          DEC     R11             %JNZRO  D.WAIT  ; WAIT FOR 20 MSECS
002074                                  ASSUME  D.ZATT,EQ,.                     ; MAKE SURE D.ZATT IS NEXT

                        ; ***   CLEAR DATT BIT FOR ATTENTION AND GET STATUS ***
                        D.ZATT:;NOP                     %JMP    D.ZATT  ; EXECUTED IN JUMP TABLE
002074  010555  007044  133605          ADD     #SDI.S1,DBAR\N,BAR %CALL S.LD13 ; [mjt08] get drive status
002075  014553  000200  000000          BIT     #DRV.OA,R13                     ; [mjt08] is the already online bit set?
002076  033753  010372  142102          MOV\F   #250.,R13       %JZRO   2$      ; [mjt08] if not,exit/r13 set for 1/4 millisecond loop
002077  033712  010002  152102  1$:     MOV\F   (RTDS),R12      %JZRO   2$      ; [mjt08] get rtds if counter not zero
002100  014552  140100  112102          BIT     #AVAIL,R12      %JDSER  2$      ; [mjt08] if bad rtds, exit/is avail bit still set?
002101  031453  010013  002077          DEC     R13,R13         %JNZRO  1$      ; [mjt08] if set,loop and decr count
                        2$:
002102  133753  000001  132011          MOV     #DATT,R13       %CALL   D.CLRS  ; R13=BIT TO CLR/GO CLR SDI STATUS
002103  033752  000200  102022          MOV     #XCMP,R12       %JMP    D.DONE  ; R12=XCMP CODE/EXIT

                        ; ***   ROUTINE TO MOVE R16 WORDS FROM @R14 TO @R12 ***
002104  033776  010004  167777  D.SMVX: MOV\T   #4,R16          %RZRO           ; IF DONE THEN R16=4/RETURN
002105  120354  007014  133605  D.SMOV: INCB    R14\0,R14,BAR   %CALL   S.LD13  ; R13=WORD FROM SOURCE
002106  120352  007012  123616          INCB    R12\0,R12,BAR   %CALL   S.ST13  ; STORE IN DESTINATION
002107  031456  000016  102104          DEC     R16             %JMP    D.SMVX  ; DECR R16 AND LOOP
                                        .PAGE
```

```
                                ; ROUTINE NAME:
                                ;       D.TIMR (SET SDI.TM AND VECT AND SUSP IN SDI.ST)
                                ;       D.TIME (SET SDI.TM)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SET THE DRIVE RESPONSE TIMER FOR THE SDI CONTROL
                                ; BLOCK POINTED TO BY DBAR. IT WILL ALSO SET THE VECTOR STATE ACTIVE
                                ; BIT IN THE SDI STATUS WORD.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO ACTIVE CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       R11 IS USED AS A TEMPORARY REGISTER
                                ;       R12 IS USED AS A TEMPORARY REGISTER
                                ;
                                ;;-

002110  013440  000000  123572  D.TIMR: NOP                     %CALL   D.GMST  ; R11=MUTEXED SDI STATUS
002111  033551  003050  133570          BIS     #<VECT+SUSP>,R11,BUF %CALL UNLOCK; INDICATE VECTOR LEVEL ACTIVE,SUSPEND
002112  010555  007033  133603  D.TIME: ADD     #SDI.TO,DBAR\N,BAR %CALL S.LD11 ; R11=DRIVE RESPONSE TIMEOUT
002113  034551  000017  010000  DTIMEA: AND     #LONIB,R11                      ; ISOLATE RESPONSE TIMEOUT
002114  033771  010002  152115          MOV\T   #2,R11          %TZRO           ; [U52EC2]IF EQ 0 THEN DEFAULT TIMEOUT=4
002115  033752  000001  123574          MOV     #1,R12          %CALL   S.ROTL  ; [U52EC2]CONVERT FROM LOG BASE 2
002116  130452  000012  133567          INC     R12             %CALL   P.LOCK  ; ADD 1 TO BE SAFE/LOCK OUT U.PROC
002117  010555  007004  133615          ADD     #SDI.TM,DBAR\N,BAR %CALL S.ST12 ; TURN ON TIMER
002120  133544  000200  137777          BIS     #PLOCK,RLL      %RET            ; FREE U.PROC/RETURN

                                ; ***   SET LONG TIMEOUT ENTRY POINT ***
002121  010555  007034  000000  D.TIML: ADD     #SDI.LT,DBAR\N,BAR              ; [U52EC2]BAR=PTR TO LONG TIMEOUT
002122  033711  000003  102113          MOV     (BUF),R11       %JMP    DTIMEA  ; [U52EC2]R11=LONG T/O,GO SET TIMER
                                        .PAGE
```



```
                                        .SBTTL  D.PROC WRITE ROUTINE (*** NEW SDI TIMING ***)
                                ;;+     09-JULY-83      UDA52 MICROCODE
                                ; ROUTINE NAME:
                                ;       D.WRIT
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL WRITE DATA FROM SECTOR BUFFER(S) ASSIGNED
                                ; TO THE SDI INTERCONNECT CONTROL BLOCK.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       DATA WRITTEN TO DISK
                                ;       ALL D.PROC REGISTERS ARE USED
                                ;
                                ;;-

002123  013440  000000  132475  D.WRIT: NOP                     %CALL   D.BFCK  ; GO CHECK BUFFER STATUS
002124  013451  010611  102310          TST     R11     @RDPF   %JZRO   D.NOBF  ; IF NONE THEN EXIT
                                                                                ;[E121]
                                ; ***   ALERT U.PROC FOR POTENTIAL EMPTY BUFFERS AS A RESULT OF THE WRITE ***    ;[E121]
                                                                                ;[E121]
002125                                  ASSUME  PKIP,EQ,BIT00                   ; MAKE SURE PKIP IS LSB         ;[E121]
002125  033551  043100  152305          BIS\F   #BFSV,R11,BUF   %JNLSB  D.IOEX  ; IF PKT NOT ACTIVE THEN DONE ELSE REQ BUFFER SERV;[E121]
002126  133544  000200  010000          BIS     #PLOCK,RLL                      ; UNLOCK U.PROC                 ;[E121]
002127  037156  000016  122520          XNOR    R16,R16         %CALL   D.WSAF  ; Wait for it to be safe to xfer        ;[E121]
002130  033751  000002  122513  D.WRTX: MOV     #2,R11          %CALL   D.WSIG  ; Verify drive is ready to xfer         ;[E121]
002131  013440  010000  112333          NOP                     %JZRO   D.RRWE  ; [16K]IF ERROR THEN GO FIND OUT WHICH ONE
002132  013740  005001  132465          MOV     #CONT,RTCS      %CALL   D.IOPW  ; GO LOAD DBAR,HDLMT
002133  033714  000523  010000  D.WRTB: MOV     (BUF),R14 @SWM                  ; [16K]R14=BUF.ST/SET WRITE MODE
002134                                  ASSUME  BGRUP,EQ,BIT00                  ; [16K]MAKE SURE BGRUP IS LSB
002134  010577  047005  152140          ADD\T   #BUF.TA,R17\N,BAR %JNLSB D.WRTC ; [16K]IF NO GROUP SELECT THEN CONT
002135  013440  000000  132535          NOP                     %CALL   D.SGRP  ; [16K]GO ISSUE SELECT GROUP COMMAND
002136  010557  017001  012335          ADD     #BUF.ST,R17\N,BAR %JNZRO D.RRDE ; [16K]IF NEQ 0 THEN SELECT GROUP ERROR
002137  013712  000002  112537          MOV     (RTDS),R12\N    %JMP    D.GRPC  ; [16K]GO DO GROUP CLEANUP
                                                                                ;[E121]
                                ; ***   HEADER SEARCH LOOP ***                  ;[E121]
                                                                                ;[E121]
002140  033712  000443  123014  D.WRTC: MOV     (BUF),R12 @SCMD %CALL   HDRCMP  ; [16K]R12=RTC,SET CMD MODE,GO DO HEADER COMPARE
002141                                  ASSUME  CONT,EQ,1                       ; MAKE SURE INCR OF R16(2) IS 3, I.E. CONT+WRT
002141  110460  015516  152146          INC\T   R16,RTCS @RSE    %JZRO  D.WRTE  ; [16K]IF HDR MATCH THEN SET WRITE GATE AND WRITE MODE
                                ; ***   HEADER NOT FOUND (MSB=HEADER SYNC T/O, LSB=LEV 1 SEND ERR, 2=RECVR RDY/R/W RDY ERR) ***
002142  013440  000500  133064          NOP             @RSE    %CALL   HDRANL  ;   ELSE RESET SERDES ENABLE/GO DO HEADER ANALYSIS;[E121]
002143  033711  010522  042352          MOV\F   (RTDS),R11 @SWM %JNZRO  D.HCER  ; [ECO#2]IF HEADER COMPARE ERROR GO DISPATCH ON IT
                                                                                ; [ECO#2]R11=DRIVE STATE
002144                                  ASSUME  RWRDY,EQ,BIT15                  ; MAKE SURE RWRDY IS MSB
002144  010577  037005  142140          ADD\T   #BUF.TA,R17\N,BAR %JMSB D.WRTC  ; IF HI THEN CONTINUE (BAR=R/T CMD PTR)
002145  013440  000000  112331          NOP                     %JMP    D.RWRE  ;   ELSE R/W READY FAILURE ERROR
                                                                                ;[E121]
                                ; ***   DESIRED HEADER FOUND - SET UP FOR WRITE ***                             ;[E121]
                                                                                ;[E121]
002146  073751  007002  000000  D.WRTE: MOV\L   #DMREG2,R11,BAR                  ; [16K]R11=4,BAR=PTR TO SECTOR SIZE
002147                                  ASSUME  DMREG4,EQ,DMREG2*2              ; [16K]MAKE SURE THIS WILL WORK
```

```
002147  033716  000523  000000          MOV     (BUF),R16 @SWM                  ; [16K]R16=SECTOR SIZE,SET WRITE MODE
002150  120356  007651  010000          INCB    R11\0,R16,BAR @RCMD             ; [16K]BAR=PTR TO # WRDS IN DATA PRE,RESET CMD MODE
                                                                                                                ;[E121]
                                ; ***   WRITE DATA PREAMBLE, SYNC, AND DATA ***                                  ;[E121]
                                                                                                                ;[E121]
002151  033711  000703  132570          MOV     (BUF),R11 @SSE  %CALL   D.WDAT  ; [16K]R11=DATA PREAMBLE,ENABLE SERDES,GO WRT DATA HDR,SYN
002152  014440  106771  052152          CLR\F   R11,SD @SECCT   %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#2(259)
002153  033456  000013  010000          MOV     R13,R16                         ; [16K]R16=BUFFER TO RELEASE
002154  034451  106011  052154          CLR\F   R11,R11,SD      %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#3(260)
002155  033453  000017  000000          MOV     R17,R13                         ; [16K]SAVE R17 IN R13 FOR LATER USE
002156  034451  106011  042156          CLR\F   R11,R11,SD      %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#4(261)
                                ; ***   RELOAD SDI CONTROL BLOCK POINTER (DBAR) ***
002157  010557  007006  000000          ADD     #BUF.SD,R17\N,BAR               ; BAR=PTR TO SDI CTRL BLK
002160  033715  106003  052160          MOV\F   (BUF),DBAR,SD   %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#5(262)
                                ; ***   RESET SEARCH LIMIT FOR HDRCMP ***
002161  010555  007001  000000          ADD     #SDI.SL,DBAR\N,BAR              ; BAR=PTR TO HDR SEARCH LIMIT
002162  033712  106003  052162          MOV\F   (BUF),R12,SD    %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#6(263)
002163  013740  007136  000000          MOV     #HDLMT,BAR                      ; BAR=PTR TO HDR LIMIT SAVE AREA
002164  033751  106100  052164          MOV\F   #BFSV,R11,SD    %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#7(264)
002165  013440  003012  000000          MOV     R12,BUF                         ; RESET HEADER SEARCH LIMIT
002166  033454  106015  042166          MOV\F   DBAR,R14,SD     %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#8(265)
002167  010555  007000  010000          ADD     #SDI.ST,DBAR\N,BAR              ; BAR=PTR TO SDI STATUS
002170  050144  106004  042170          ADD\RF  RLL,RLL,SD      %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#9(266)
002171  033712  020003  052176          MOV\F   (BUF),R12       %JNCRY  D.WRTF  ; IF LOCKED THEN CONTINUE
002172  013440  106012  042172          MOV\F   R12,SD          %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#10(267)
002173                                  ASSUME  PKIP,EQ,BIT00                   ; MAKE SURE PKIP IS LSB
002173  033172  043011  042174          BIS\T   R11,R12,BUF     %TLSB           ; IF PKT STILL ACTIVE THEN SET BFSV IN SDI.ST
002174  133544  106200  052174          BIS\F   #PLOCK,RLL,SD   %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#11(268)
002175  010557  007000  102201          ADD     #BUF.NL,R17\N,BAR %JMP  D.WRTG  ; BAR=PTR TO NXT BUFFER/CONTINUE
002176  014440  106011  042176  D.WRTF: CLR\F   R11,SD          %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#11(268)
002177  014440  106011  052177          CLR\F   R11,SD          %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#11(268)
002200  010557  007000  000000          ADD     #BUF.NL,R17\N,BAR               ; BAR=PTR TO NXT BUFFER/CONTINUE
                                ; ***   IMPORTANT - RESET ECC TIME (@RECCT) ON DATA WORD 269, WORDS 269, 270 &271 MUST BE ZERO ***
002201  014440  106571  042201  D.WRTG: CLR\F   R11,SD  @RECCT  %JNDCLK .       ; WHEN DRV CLK,(SD)=(269)=0/RECCT
002202  033717  000003  000000          MOV     (BUF),R17                       ; R17=NEXT BUFFER POINTER
                                ; ***   FORM NEXT DATA BUFFER ADDRESS IN DBAR ***
002203  034451  106011  042203          CLR\F   R11,R11,SD      %JNDCLK .       ; [16K]WHEN DRV CLK,(SD)=WORD 270=0
002204  010557  167002  112323          ADD     #BUF.BP,R17\N,BAR %JLATE D.OVER ; [ECO#2]BAR=PTR TO START OF SECTOR DATA
002205  034452  106012  042205          CLR\F   R12,R12,SD      %JNDCLK .       ; [16K]WHEN DRV CLK,(SD)=WORD 271=0
002206  033715  140003  112327          MOV     (BUF),DBAR      %JDSER  D.RTDE  ; [16K]DBAR=DATA BUFFER POINTER
002207                                  ASSUME  CONT,EQ,1                       ; MAKE SURE CONT IS 1
002207  110440  105551  052207          INC\F   R11,RTCS @RECC  %JNDCLK .       ; [16K]WHEN DRV CLK,@RECC,DROP WRT GATE/WORD=272
002210  010553  007000  010000          ADD     #BUF.NL,R13\N,BAR               ; [16K]BAR=PTR TO LAST BUFFER STATUS
002211  037151  000511  122513          XNOR    R11,R11 @RSE    %CALL   D.WSIG  ; [16K]RESET SERDES,GO WAIT FOR RECVR RDY & R/W RDY
002212  115557  013100  152333          BIC\F   #BFULL,R17\N,BUF %JZRO  D.RRWE  ; [16K]IF ERROR THEN PROCESS/ELSE CLR FULL FLG
002213  030554  037007  033622          ADD     #SDI.DB,R14,BAR %CNMSB  S.ST17  ; IF NOT LAST THEN UPDATE SDI.DB
                                ; ***   CHECK FOR DM MODE ***
002214  114544  000100  010000          BIT     #DMODE,RLL                      ; [16K]IF IN DM MODE
002215  010553  017002  002223          ADD     #BUF.BP,R13\N,BAR %JNZRO D.WRTI ; [16K]  THEN SKIP BUFFER RELEASE
002216  013740  003000  000000          MOV     #0,BUF                          ; [16K]INDICATE BUFFER RELEASED
                                ; ***   RELEASE DATA BUFFER IN R16 [16K] ***
002217  025544  000001  010000          BIC     #1,RLL\0                        ; [16K]IF BUFFER LOCK CLEAR
002220  010544  047003  142217          ADD\F   #3,RLL\N,BAR    %JNLSB  .-1     ; [16K]  THEN WAIT
002221  013440  003016  010000          MOV     R16,BUF                         ; [16K]  ELSE RETURN BUFFER TO STACK
```



```
002222  030544  000003  010000          ADD     #3,RLL                          ; [16K]RELEASE LOCK/UPDATE STACK PTR
002223  010557  007000  000000  D.WRTI: ADD     #BUF.NL,R17\N,BAR               ; BAR=PTR TO NEXT BUFFER STATUS
002224                                  ASSUME  BLAST,EQ,BIT15                  ; MAKE SURE BLAST IS MSB
002224  033714  030003  002227          MOV     (BUF),R14       %JNMSB  D.WRTJ  ; [16K][U52EC1]IF NOT LAST BUFFER THEN EXIT, SETTING UP R1
002225  013740  007236  133625          MOV     #BTCNT,BAR      %CALL   INITM1  ; [U52EC1]FORCE U.PROC TO LOOK FOR WORK TO DO
002226  013440  000000  102231          NOP                     %JMP    D.IODN  ; [U52EC1]CONTINUE
002227  114554  000100  000000  D.WRTJ: BIT     #BFULL,R14                      ; [U52EC1]IF NXT BUFFER FULL
002230  010557  017001  002133          ADD     #BUF.ST,R17\N,BAR %JNZRO D.WRTB ;   [16K]THEN WRITE ANOTHER SECTOR
                                                                                                                ;[E121]
                                ; ***   I/O DONE - DM MODE REQUEST OR NEED BUFFER FILL SERVICE FROM U.PROC ***   ;[E121]
                                                                                                                ;[E121]
002231  034451  000551  010000  D.IODN: CLR     R11     @RECC                   ; ZAP R11/RESET ECC
002232  114544  000100  010000          BIT     #DMODE,RLL                      ; IF IN DM MODE
002233  034453  010453  013454          CLR     R13     @SCMD   %JNZRO  XFCSST  ;   THEN RETURN TO DM
002234  010557  007006  133607          ADD     #BUF.SD,R17\N,BAR %CALL S.LDDB  ; DBAR=SDI CONTROL BLK PTR
002235  010555  007001  133604          ADD     #SDI.SL,DBAR\N,BAR %CALL S.LD12 ; R12=SEARCH LIMIT
002236  055552  000001  010000          BIC\R   #1,R12                          ; DIVIDE BY 2
002237  010555  007023  133615          ADD     #SDI.RO,DBAR\N,BAR %CALL S.ST12 ; RESET ROTATIONAL OPTIMIZATION WORD
002240  010555  007006  133603          ADD     #SDI.UB,DBAR\N,BAR %CALL S.LD11 ; R11=U.PROC BUFFER CNTRL BLK POINTER
                                ; ***   R16 = POINTER OF BUFFER CONTROL BLOCK JUST READ ***
002241  036151  000016  000000          XOR     R16,R11                         ; CHECK IF SAME AS CURRENT PTR
002242                                  ASSUME  BLAST,EQ,BIT15                  ; MAKE SURE BLAST IS MSB
002242  135551  030300  002251          BIC     #<BLAST!BFULL>,R11 %JNMSB D.IODA; [16K]IF R17 IS NOT LAST THEN CONTINUE
002243  133553  010100  042251          BIS\F   #ERRIP,R13      %JNZRO  D.IODA  ; [EERREC]IF SDI.UB EQ SDI.DB THEN CLR ERRIP
                                ;       BECAUSE IF BLAST SET AND CURRENT BCB PTR EQ SDI.UB THEN IT HAS TO BE A
                                ;       U.PROC RECOVERY OPERATION AND WE MUST CLEAR ERRIP TO CLEAN UP THE STATE
002244  010556  007001  123603          ADD     #BUF.ST,R16\N,BAR %CALL S.LD11  ; [EERREC]GET BUFFER STATUS WORD [rae09]
002245  014551  000040  010000          BIT     #BRCTS,R11                      ; [mjt09]is brcts set?
002246  033571  010020  042247          BIS\T   #BRARS,R11      %TNZRO          ; [mjt09][EERREC]SET READ AFTER RCT SEARCHED FLAG
002247  135551  000001  000000          BIC     #BERDN,R11                      ; [EERREC]CLR ERROR RECOVERY CMD FLAG
002250  035551  003040  010000          BIC     #BRCTS,R11,BUF                  ; [EERREC]CLEAR BIT FOR D PROC TO AVOID ENDLESS LOOP
002251  033752  000140  102265  D.IODA: MOV     #<SUSP+BFSV>,R12 %JMP   D.IODC  ; R12=SUSPEND & BUFFER SERVICE
                                                                                                                ;[E121]
                                ; ***   FATAL I/O ERROR ENTRY POINT ***                                          ;[E121]
                                                                                                                ;[E121]
002252  033751  000002  122043  D.IODB: MOV     #FM.DSK,R11     %CALL   D.LOGF  ; [U52EC3]R11=LOG CODE/GO SET FATAL LOG
002253  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=SDI STATUS
002254                                  ASSUME  RVCT,EQ,BIT15                   ; [U52EC2]MAKE SURE RVCT IS MSB
002254  013440  030000  112265          NOP                     %JMSB   D.IODC  ; [U52EC2]IF FATAL ERROR AND REVECT ACTIVE THEN CONT
002255  010557  007004  133606          ADD     #BUF.HH,R17\N,BAR %CALL S.LD14  ; R14=ERROR LBN HI
002256                                  ASSUME  XBNCOD&BIT14,EQ,0               ; [U52EC3]MAKE SURE BIT14
002256                                  ASSUME  RBNCOD&BIT14,NE,0               ; [U52EC3]CHK FILTERS RBN'S
002256  114554  000100  000000          BIT     #BIT14,R14                      ; [U52EC3][U52EC2]IF HDR CODE EQ PRIM RBN
002257  133553  010100  032460          BIS     #ERRIP,R13      %CNZRO  D.RBLH  ; [U52EC2]  THEN RESTORE BUF.HL/BUF.HH
002260  010555  007022  133603          ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD11 ; R11=MSCP PKT PTR
002261  010551  007017  123617          ADD     #S.LBNH,R11\N,BAR %CALL S.ST14  ; SAVE AS FINAL LBN HI
002262  010557  007003  123606          ADD     #BUF.HL,R17\N,BAR %CALL S.LD14  ; R14=ERROR LBN LO
002263  010551  007016  133617          ADD     #S.LBNL,R11\N,BAR %CALL S.ST14  ; SAVE AS FINAL LBN LO
002264  010557  007000  123606          ADD     #BUF.NL,R17\N,BAR %CALL S.LD14  ; R14=CURRENT BUFFER STATUS
002265  010555  007020  133623  D.IODC: ADD     #SDI.EO,DBAR\N,BAR %CALL S.CLRB ; CLEAR LEVEL 0 ERROR STATE
002266  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11 = SDI STATUS
002267                                  ASSUME  RVCT,EQ,BIT15                   ; MAKE SURE RVCT IS MSB
002267  013440  030000  002272          NOP                     %JNMSB  D.IODD  ; [U52EC2]IF REVECTORING NOT ACTIVE THEN CONTINUE
002270  033152  000011  122011          OR      R11,R12         %CALL   D.CLRS  ; [U52EC2]  ELSE CLEAR STATUS BITS IN R13
```

```
002271  014552  000004  103335          BIT     #DERR,R12       %JMP    D.RV4A  ; TEST FOR ERRORS/RESUME REVECTOR PROCESS
                                ; ***   THE FOLLOWING INTERLOCKED BUFFER STATUS CHECK IS MADE TO AVOID THE RACE
                                ; ***   CONDITION WHICH OCCURS WHEN THE U.PROC HAS SET THE BUFFER STATUS
                                ; ***   AFTER THE D.PROC HAS EXAMINED IT, CLEARS SDI STATUS BITS 'SUSP' AND
                                ; ***   'BFSV', THEN THE D.PROC SETS 'BFSV' AND 'SUSP' THEREBY LOCKING ITSELF
                                ; ***   OUT BECAUSE THE D.PROC WILL ONLY ENTER D.READ AND D.WRIT WHEN 'SUSP'
                                ; ***   IS CLEAR !
002272  010557  007000  133567  D.IODD: ADD     #BUF.NL,R17\N,BAR %CALL P.LOCK  ; BAR=BUFFER STATUS PTR/LOCK OUT U.PROC
002273  036514  000003  122306          XOR     (BUF),R14       %CALL   D.CLUR  ; [U52EC2]IF PREVIOUS AND INTERLOCKED STATUS DIFFER
                                                                                ; [U52EC2]CLEAR URETRY IN SDI.ES
002274  134554  000100  010000          AND     #BFULL,R14                      ; ISOLATE BUFFER FULL/EMPTY FLAG
002275  035572  010140  042276          BIC\T   #<SUSP+BFSV>,R12 %TNZRO         ;   THEN DON'T SET SUSPEND AND BUFFER SERVICE BITS
002276  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; R11=SDI STATUS (MUTEXED)
002277  014551  000202  000000          BIT     #<XCMP!SEEK>,R11                ; [U52EC1]IF SEEK OR DONE
002300  013440  010000  002305          NOP                     %JNZRO  D.IOEX  ; [U52EC1]  THEN JUST FREE U.PROC/EXIT
002301  013451  000011  102303          TST     R11             %JMP    D.SETA  ; CHK IF PKT STILL ACTIVE               ;[E121]
                                ; ***   ROUTINE TO CLEAR SDI STATUS BITS IN R13 AND SET BITS IN R12 ***
002302  013440  000000  123572  D.SETS: NOP                     %CALL   D.GMST  ; R11=MUTEXED SDI STATUS
002303                          ASSUME  PKIP,EQ,BIT00                           ; MAKE SURE PKIP IS LSB
002303  035151  040013  142305  D.SETA: BIC\F   R13,R11          %JNLSB  D.IOEX  ; IF PKT NOT ACTIVE THEN DONE ELSE ZAP FLAGS
002304  033151  003012  010000          BIS     R12,R11,BUF                     ; SET STATUS FLAGS FROM R12
                                ; ***   I/O EXIT - RELEASE U.PROC, GO IDLE ***
002305  133544  000200  100701  D.IOEX: BIS     #PLOCK,RLL      %JMP    D.IDLE  ; UNLOCK UPROC/GO IDLE

                                ; ***   ROUTINE TO CLEAR URETRY IN SDI.ES, USED BY D.IODD & D.RVC8 ***
002306  010555  007026  123603  D.CLUR: ADD     #SDI.ES,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=EXTENDED STATUS
002307  035551  003040  137777          BIC     #URETRY,R11,BUF %RET            ; [U52EC2]CLR URETRY,RESET,RETURN
                                                                                                                ;[E121]
                                                                                                                ;[E121]
                                ;       ENTER HERE FROM D.READ/D.WRIT WHEN THE FIRST BUFFER IS FULL/EMPTY        ;[E121]
                                                                                                                ;[E121]
002310                          D.NOBF: ASSUME  PKIP,EQ,BIT00                   ;enter with interlocked SDI.ST in CC's  ;[E121]
002310  033151  043012  152305          BIS\F   R12,R11,BUF     %JNLSB  D.IOEX  ;exit if no pkt in prog, else           ;[E121]
                                                                                ;set SUSP and BFSV in SDI.ST            ;[E121]
002311  010557  007001  133606          ADD     #BUF.ST,R17\N,BAR %CALL S.LD14  ;get buffer status in R14               ;[E121]
002312                          ASSUME  BGRUP,EQ,BIT00                          ;to see if group sel needed             ;[E121]
002312  133544  040200  110701          BIS     #PLOCK,RLL      %JNLSB  D.IDLE  ;release interlock, leave if no grp sel ;[E121]
002313  013440  000000  132535          NOP                     %CALL   D.SGRP  ;send grp sel to drive, do bookkeeping  ;[E121]
002314  010557  017001  012335          ADD     #BUF.ST,R17\N,BAR %JNZRO D.RRDE ;jump if error sending grp sel          ;[E121]
002315  133752  000002  010000          MOV     #1000,R12                       ;Avoid waiting infinitely below         ;[E121]
002316                          ASSUME  RWRDY,EQ,BIT15                          ;                                       ;[E121]
002316  013712  010002  102320  10$:    MOV     (RTDS),R12\N    %JZRO   20$      ;wait until Read/Write Ready drops      ;[E121]
002317  031452  030012  112316          DEC     R12             %JMSB   10$     ;so we won't be fooled later            ;[E121]
002320  050144  000004  133566  20$:    ADD\R   RLL,RLL         %CALL   P.LCKA  ;get interprocessor interlock again     ;[E121]
002321  033712  000003  000000          MOV     (BUF),R12                       ;clear the "group select needed"        ;[E121]
002322  035552  003001  112305          BIC     #BGRUP,R12,BUF  %JMP    D.IOEX  ;under interlock to avoid race          ;[E121]
                                                                                ;with upper proc trying to change       ;[E121]
                                                                                ;BFULL bit, and exit                    ;[E121]
                                .PAGE
```



```
                                ;;+
                                ; PROCESS NAME:
                                ;       READ/WRITE ERROR HANDLING
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       EVERY POSSIBLE ERROR THAT CAN OCCUR DURING A READ OR WRITE OPERATION
                                ; WILL BE DIRECTED THROUGH THIS CODE.
                                ;
                                ; INPUTS:
                                ;       R17             BUFFER POINTER
                                ;
                                ; OUTPUTS:
                                ;       VARY ACCORDING TO ERROR TYPE
                                ;
                                ;;-

                                ; ***   ENTRY POINT FOR SERDES LATE (OVERRUN) ERRORS ***
002323  033756  000052  102332  D.OVER: MOV     #<ST.CNT+SC.OVR>,R16 %JMP D.RWRA ; R16=SERDES OVERRUN ERROR

                                ; ***   DRIVE PROCESSOR INTRUCTION TIMEOUT RECOVERY ROUTINE ***
                                ; ***   NOTE - WHEN ENTERING AT D.TIMO, BIT TEST FOR OPERATION MODE IN RLL
002324  033751  010015  027777  D.TIMO: MOV     #ER.TMO,R11     %RNZRO          ; [16K]IF DIAGNOSTIC TIMEOUT THEN RETURN
002325  033756  000253  010000          MOV     #<ST.DRV!SC.DCL>,R16            ; [ECO#1]R16=MSCP STATUS
002326                          ASSUME  D.TOER,EQ,.                             ; FALL INTO D.TOER

                                ; ***   ENTRY POINT FOR I/O INSTRUCTION TIMEOUT ***
002326  133754  000020  102424  D.TOER: MOV     #IOCLK,R14      %JMP    D.IOER  ; R14=CLK ERROR CODE/CONTINUE

                                ; ***   ENTRY POINT FOR RTDS RECEIVE ERROR DURING READ/WRITE OPERATION ***
002327  033756  000013  010000  D.RTDE: MOV     #ST.DRV,R16                     ; [ECO#1]R16=DRIVE ERROR CODE
002330  133556  000001  102332          BIS     #SC.LVO,R16     %JMP    D.RWRA  ; [ECO#1]SET R14 AND CONTINUE

                                ; ***   ENTRY POINT FOR READ/WRITE READY FAILURE ***
002331  033756  000213  000000  D.RWRE: MOV     #<ST.DRV+SC.RWR>,R16            ; R16=READ/WRITE RDY FAILURE ERROR
002332  133754  000040  102424  D.RWRA: MOV     #IORWR,R14      %JMP    D.IOER  ; R14=RD/WRT RDY ERROR CODE/CONTINUE

                                ; ***   ENTRY POINT FOR RECEIVER READY OR READ/WRITE READY FAILURE ***
002333  033714  000002  010000  D.RRWE: MOV     (RTDS),R14                      ; IF RD/WRT RDY ERROR
002334                          ASSUME  RWRDY,EQ,BIT15                          ; MAKE SURE RWRDY IS MSB
002334  013440  030000  002331          NOP                     %JNMSB  D.RWRE  ; IF NOT HI THEN PROCESS

                                ; ***   ENTRY POINT FOR RECEIVER READY TIMEOUT ERROR ***
002335  033756  000313  102326  D.RRDE: MOV     #<ST.DRV+SC.RRD>,R16 %JMP D.TOER ; R16=RECEIVER READY FAILURE ERROR

                                ; ***   ENTRY POINT FOR SUSPECTED POSITIONER ERROR
002336  033756  000153  010000  D.TKER: MOV     #<ST.DRV+SC.POS>,R16            ; R16=POSITIONER ERROR
002337  133754  000010  102424  D.TKEA: MOV     #IOSEK,R14      %JMP    D.IOER  ; R14=SEEK ERROR/CONTINUE

                                ; ***   ENTRY POINT FOR HEADER ERROR AFTER RCT SEARCHED
002340  034456  000016  112343  D.RCTE: CLR     R16             %JMP    D.EERR  ; [EERREC] R16=0 , GO INTO ENHANCED ERROR RECOVERY
                                ; ***   ENTRY FOR DATA SYNC TIMEOUT
002341  133756  000004  112343  D.DTER: MOV     #BDSNF,R16      %JMP    D.EERR  ; [EERREC] R16=DSNF FLAG, GO INTO ENHANCED ERROR RECOVER
                                ; ***   ENTRY FOR LOSS READ/WRITE READY ERROR
002342  133756  000002  000000  D.LRER: MOV     #BLRWR,R16                      ; [EERREC] R16=LRWR FLAG, GO INTO ENHANCED ERROR RECOVER
```

```
002343  010557  007006  133607  D.EERR: ADD     #BUF.SD,R17\N,BAR %CALL S.LDDB  ; [EERREC] RESTORE DBAR
002344  010555  007007  123611          ADD     #SDI.DB,DBAR\N,BAR %CALL S.LD17 ; [EERREC] R17 = BUFFER CNTR BLK PTR
002345  010557  007001  133606          ADD     #BUF.ST,R17\N,BAR %CALL S.LD14  ; [EERREC] GET SDI STATUS
002346  033154  003016  000000          BIS     R16,R14,BUF                     ; [EERREC] SET ERROR FLAG
002347  010557  007000  123606          ADD     #BUF.NL,R17\N,BAR %CALL S.LD14  ; [EERREC] GET PNT TO NEXT DATA BUFFER
002350  133554  003100  000000          BIS     #BFULL,R14,BUF                  ; [EERREC] SET FULL
002351  037156  000016  102231          XNOR    R16,R16         %JMP    D.IODN  ; [EERREC] R16 = 177777Q FOR D.IODN AND EXIT

                                        .PAGE
```



```
                                ; ***   ENTRY POINT FOR HEADER COMPARE ERROR ***
                                ; ***   THIS CODE PROCESSES ALL HEADER COMPARE ERRORS (LBN,RBN,RCT,FCT) ***
002352  010557  007001  133605  D.HCER: ADD     #BUF.ST,R17\N,BAR %CALL S.LD13  ; [16K][ECO#2]R13=ON CYL GOODNESS FLAG
002353  033456  000011  000000          MOV     R11,R16                         ; [ECO#2]R16=ERROR CODE FOR DM
002354  114544  030100  100701          BIT     #DMODE,RLL      %JNEG   D.IDLE  ; IF WE ARE TOO FAR AWAY THEN GO IDLE
002355  131551  010002  052424          SUB\F   #2,R11          %JNZRO  D.IOER  ; IF DM MODE THEN SKIP/CONTINUE
002356  016551  010004  112335          XOR     #4,R11\N        %JZRO   D.RRDE  ; IF I/O ERROR FOUND DURING HDR RD THEN PROCESS
002357  111551  010002  102361          SUB     #2,R11\N        %JZRO   D.HCEO  ; [U52EC2][ECO#2]IF RBN/XBN HEADER PROBLEM THEN DONE
002360  114573  010010  152361          BIT\T   #BGOOD,R13      %TZRO           ; [ECO#2]IF SEARCH LIMIT EXCEEDED THEN TEST BGOOD
002361  135553  013010  102336  D.HCEO: BIC     #BGOOD,R13,BUF  %JZRO   D.TKER  ; [U52EC2][ECO#2]IF BGOOD EQ 0 THEN POSITIONER ERROR
002362  010557  007006  133607          ADD     #BUF.SD,R17\N,BAR %CALL S.LDDB  ; RESTORE SDI CTL BLK PTR
                                ; ***   HEADER NOT FOUND, LET'S CHECK IT FOR RCT LBN OR FCT BN ***
002363  013440  000000  123053          NOP                     %CALL   D.LDHD  ; [U52EC2]R13/R14=LO/HI HEADER
002364  010555  007067  133604          ADD     #SDI.RT,DBAR\N,BAR %CALL S.LD12 ; Get RBN's/track                       ;[E121]
002365  034552  000077  010000          AND     #RBNMSK,R12                     ; Mask out crap                         ;[E121]
002366  033452  010014  102372          MOV     R14,R12         %JZRO   10$     ; If no RBN'S, no revectoring!          ;[E121]
                                                                                ; Else copy header for RBN test         ;[E121]
002367  134552  000360  000000          AND     #HDCOD,R12                      ; Isolate header code in hi hdr         ;[E121]
002370  136552  000140  010000          XOR     #RBNCOD,R12                     ; RBN's can be revectored; all other    ;[E121]
002371  013440  010000  112401          NOP                     %JZRO   D.HCEA  ; non-LBN codes will fail RCT test      ;[E125] mjt
002372  010555  007076  133604  10$:    ADD     #SDI.L2,DBAR\N,BAR %CALL S.LD12 ; [U52EC2]R12=1ST RCT LBN HI            ;[E121]
002373  010555  007075  123603          ADD     #SDI.L1,DBAR\N,BAR %CALL S.LD11 ;[V05] [U52EC2]R11=1ST RCT LBN LO
002374  112152  000014  000000          CMP     R14,R12                         ; [U52EC2]COMPARE HI BLK NUMBERS
002375  112171  010013  152376          CMP\T   R13,R11         %TZRO           ; [U52EC2]IF EQ 0 THEN COMPARE LO BLK NUMBERS
002376  033753  020100  012401          MOV     #BFSV,R13       %JNCRY  D.HCEA  ; [U52EC3][U52EC2]IF NOT IN RCT SPACE THEN OK
                                                                                ; [U52EC3]TO DO REVECTORING
                                ; ***   XBN (FCT) OR RCT LBN HEADER NOT FOUND OR NO REVECTORING EXIT ***                ;[E121]
002377  033752  000244  000000          MOV     #<SUSP!DERR!XCMP>,R12           ; [U52EC3]R12=SDI.ST BITS TO SET
002400  033756  000110  112422          MOV     #<ST.DAT+SC.HDR>,R16 %JMP D.HCED; [U52EC3][U52EC2]R16=HEADER NOT FOUND ERROR
                                ; ***   BLOCK NOT IN RCT SPACE - CHECK IF REVECTORING ACTIVE ***
002401  010555  007000  133603  D.HCEA: ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]GET SDI STATUS WORD
002402                          ASSUME  RVCT,EQ,BIT15
002402  133752  030200  112415          MOV     #RVCT,R12       %JMSB   D.HCEC  ; [U52EC2]IF REVECT ON THEN CANNOT NEST IT
002403  010557  007001  133603          ADD     #BUF.ST,R17\N,BAR %CALL S.LD11  ; [EERREC]GET BUFFER STATUS WORD
002404  014551  000060  000000          BIT     #<BRCTS+BRARS>,R11              ; [EERREC]CHECK IF THIS IS DURING EHNAHCED RECOVERY
002405  033776  010110  052340          MOV\T   #<ST.DAT+SC.HDR>,R16 %JNZRO D.RCTE ; [EERREC]IF SO THEN SET STATUS AND CONTINUE WITH RECOV
002406  010557  007002  133006          ADD     #BUF.BP,R17\N,BAR %CALL D.ALBF  ; [U52EC2MUST HAVE BUFFER TO START REVECTOR!!!!!
002407  130464  010004  140701          INC\T   RLL,RLL         %JZRO   D.IDLE  ; [U52EC2]IF NO BUFFER THEN GO AWAY
                                                                                ; [U52EC2]TO PREVENT DEADLOCK !!!!
                                ; ***   NOW WE MUST FILTER OUT THE PRIMARY "HEADER NOT FOUND" CASE ***
002410  114554  000360  010000          BIT     #HDCOD,R14                      ; [U52EC2]IF PRIMARY HEADER CODE
002411  013440  010000  022460          NOP                     %CNZRO  D.RBLH  ; [U52EC2]  THEN RESTORE BUF.HL/BUF.HH
002412  010555  007100  123622          ADD     #SDI.PH,DBAR\N,BAR %CALL S.ST17 ; [V05][U52EC2]SAVE PTR OF REVECTOR BUFFER IN SDI.S7
002413  010555  007024  123623          ADD     #SDI.OE,DBAR\N,BAR %CALL S.CLRB ; [U52EC1]NO OVERLAPS ON REVECTORS
002414  034453  000013  112302          CLR     R13             %JMP    D.SETS  ; [ECO#2]GO TURN ON REVECTORING
                                ; ***   HEADER NOT FOUND AND REVECTORING ACTIVE, CAN BE RCT LBN OR RBN ! ***
002415  033756  000110  000000  D.HCEC: MOV     #<ST.DAT!SC.HDR>,R16            ; [U52EC2]R16=HDR NOT FOUND CODE
002416  034453  000013  000000          CLR     R13                             ; [U52EC2]ZAP R13
002417  010557  007004  123604          ADD     #BUF.HH,R17\N,BAR %CALL S.LD12  ; [U52EC2]R12=HI BLK NUMBER
002420  134552  000360  000000          AND     #HDCOD,R12                      ; [U52EC2]IF NOT LBN (RCT)
002421  033752  010004  002252          MOV     #DERR,R12       %JNZRO  D.IODB  ; [U52EC2]  THEN MUST BE RBN AND ERROR IS FATAL
                                                                                ; [U52EC2]R12=DRIVE ERROR FLAG
002422  010557  007001  133603  D.HCED: ADD     #BUF.ST,R17\N,BAR %CALL S.LD11  ; [U52EC2]R11=BUFFER STATUS WORD
002423  033551  003004  112252          BIS     #BNXCOP,R11,BUF %JMP    D.IODB  ; [U52EC2]INDICATE TO REVECTORING TO USE NEXT COPY
```

```
                                    ;                                                   ; [U52EC2]OF RCT IF APPROPRIATE
                                            .PAGE
```



```
                                    ; *-------------------------------------------------------------------------*
                                    ; *     EEEEEEE  RRRRRR   RRRRRR    00000   RRRRRR                        *
                                    ; *     E        R     R  R     R  O     O  R     R                       *
                                    ; *     E        R     R  R     R  O     O  R     R                       *
                                    ; *     EEEEEE   RRRRRR   RRRRRR   O     O  RRRRRR                        *
                                    ; *     E        R   R    R   R    O     O  R   R                         *
                                    ; *     E        R    R   R    R   O     O  R    R                        *
                                    ; *     EEEEEEE  R     R  R     R   00000   R     R                       *
                                    ; *-------------------------------------------------------------------------*
                                    ; ***   PLEASE NOTE THAT EVERY I/O ERROR EXITS THROUGH D.IOER ! ***

002424  013440  000500  131270  D.IOER: NOP             @RSE     %CALL   D.CLCS  ; RESET SERDES ENABLE/CLEAR UDA RTCS
002425  013740  007007  123622          MOV     #DMREGO,BAR      %CALL   S.ST17  ; SAVE R17 IN CASE DM MODE
002426  114544  000040  010000          BIT     #DMBEG,RLL                       ; IF IN DM MODE
002427  033451  010556  003454          MOV     R16,R11 @RECC    %JNZRO  XFCSST  ;   THEN RETURN/RESET ECC ENABLE
002430  010557  007006  133607          ADD     #BUF.SD,R17\N,BAR %CALL S.LDDB  ; GET POINTER TO SDI CTL BLOCK
002431  013740  067005  123607          MOV     #DMREGS,BAR      %CDPF   S.LDDB  ; IF REPLACE TIMEOUT THEN RESTORE DBAR PROPERLY
                                    ; ***   AT THIS POINT R17 HAS A BUFFER CTL BLK ADDR BUT IT MAY NOT BE CORRECT -
                                    ; ***   CERTAIN ERRORS ARE CHECKED FOR AFTER THE NEXT BUFFER CTL BLK HAS BEEN CHAINED TO
002432  010555  007007  123611          ADD     #SDI.DB,DBAR\N,BAR %CALL S.LD17 ; [U52EC2]RESTORE BUFFER CNTL BLK PTR FROM SDI.DB
002433  010555  007026  123603          ADD     #SDI.ES,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]R11=EXTENDED STATUS
002434  014551  000040  010000          BIT     #URETRY,R11                      ; [U52EC2]IF U.PROC RETRY READ,
002435  010575  017006  073611          ADD\T   #SDI.UB,DBAR\N,BAR %CNZRO S.LD17 ; [U52EC2] THEN RESTORE R17 FROM SDI.UB
                                    ; ***   CHECK FOR ERROR LOGGING ENABLED ***
002436  033753  000002  121633          MOV     #FM.DSK,R13      %CALL   C.LOGS  ; [U52EC2]GO CHECK IF LOGGING ENABLED
002437  033451  000014  133053          MOV     R14,R11          %CALL   D.LDHD  ; [ECO#2]SAVE R14,R13/R14=LO/HI LBN
002440  114554  000360  123233          BIT     #HDCOD,R14       %CALL   D.SLBN  ; [U52EC2][ECO#2]SAVE LBN/2NDARY RBN IN P.BUF2/3
002441  033753  000100  010000          MOV     #BFSV,R13                        ; [ECO#2]R13=BUFFER SERVICE FLAG
002442  010555  007000  133606          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD14 ; [U52EC2]R14=SDI STATUS
002443                                  ASSUME  RVCT,EQ,BIT15                    ; [U52EC2]MAKE SURE MSB
                                                                                 ; [U52EC2]DON'T RESET GROUP IF REVECTORING ACTIVE
002443  033454  030011  032544          MOV     R11,R14          %CNMSB  D.GGRP  ; [U52EC2][ECO#2][ECO#1]RESTORE R14 FROM R11
002444  033752  000244  000000          MOV     #<SUSP+DERR+XCMP>,R12            ; R12=FLAGS TO SET
002445  010555  007020  123603          ADD     #SDI.EO,DBAR\N,BAR %CALL S.LD11 ; R11=I/O RETRY STATE
002446  114551  000004  000000          BIT     #IOCNT,R11                       ; IF 2ND TIME AROUND
002447  114551  010074  002422          BIT     #IORTY,R11       %JNZRO  D.HCED  ; [U52EC2]IF 1ST TIME AROUND
002450  133574  010004  042451          BIS\T   #IOCNT,R14       %TNZRO          ;   THEN SET LAST INDICATOR
002451  013440  003014  132011          MOV     R14,BUF          %CALL   D.CLRS  ; SET UP I/O RETRY WORD/GO CLR BFSV FLAG (R13)
002452  114554  000020  000000          BIT     #IOCLK,R14                       ; IF INIT NOT REQUIRED
002453  033751  010255  102455          MOV     #V.ERRO&LOBYT,R11 %JZRO D.IOEA  ; [U52EC1]THEN GO START I/O RECOVERY (WITHOUT INIT)
002454  033751  000254  010000          MOV     #V.INIT&LOBYT,R11                ; [U52EC1]ELSE GO START I/O RECOVERY (WITH INIT)
002455  010555  007021  133614  D.IOEA: ADD     #SDI.SV,DBAR\N,BAR %CALL S.ST11 ; [U52EC1]SAVE STATE VECTOR
002456  013440  000000  132110          NOP                      %CALL   D.TIMR  ; [U52EC1]SET SUSP & VECT/START TIMER
002457  013440  000000  110701          NOP                      %JMP    D.IDLE  ; [U52EC1]GO IDLE

                                    ; ***   RESTORE BUF.HL/BUF.HH FROM P.BUF2/P.BUF3 ***
002460  010555  007022  133603  D.RBLH: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD11 ; [U52EC2]  ELSE R11=MSCP PKT PTR
002461  010555  007031  123606          ADD     #SDI.RL,DBAR\N,BAR %CALL S.LD14 ; [QDA][U52EC2]R14=LO SAVED ORIG LBN
002462  010557  007003  123617          ADD     #BUF.HL,R17\N,BAR %CALL S.ST14  ; [U52EC2]RESTORE ORIG LO LBN TO BUF.HL
002463  010555  007032  123606          ADD     #SDI.RH,DBAR\N,BAR %CALL S.LD14 ; [QDA][U52EC2]R14=HI SAVED ORIG LBN
002464  010557  007004  103617          ADD     #BUF.HH,R17\N,BAR %JMP  S.ST14  ; [U52EC2]RESTORE ORIG HI LBN TO BUF.HH/RETURN

                                            .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.IOPW  (I/O SETUP FOR WRITE COMMAND)
                                ;       D.IOPR  (I/O SETUP FOR READ COMMAND)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       D.IOPW WILL ADJUST THE DATA PREAMBLE (DMREG4) ACCORDING TO THE
                                ; SPEED OF THE DRIVE AND FALL INTO D.IOPR.
                                ;       D.IOPR WILL SET THE HEADER COMPARE SEARCH LIMIT (HDLMT), SET
                                ; WORDS TO TRANSFER IN R16, LOAD DBAR TO POINT TO THE START OF THE DATA
                                ; BUFFER, AND TEST BUF.NL FOR GROUP SELECT NEEDED (BGRUP).
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO SDI CONTROL BLOCK
                                ;       R17             POINTER TO CURRENT SECTOR BUFFER
                                ;       DMREG4          DATA PREAMBLE SIZE FROM DRIVE CHARACTERISTICS (D.IOPW)
                                ;
                                ; OUTPUTS:
                                ;       DBAR            POINTER TO START OF DATA FOR THIS TRANSFER
                                ;       DMREG4          ADJUSTED FOR FUDGE AND WORDS IN SERDES
                                ;       HDLMT           HEADER COMPARE SEARCH LIMIT                                     ;[E121]
                                ;       R11             0                                                               ;[E121]
                                ;
                                ;;-

002465  013740  007004  133610  D.IOPW: MOV     #DMREG4,BAR     %CALL   S.LD16  ; R16=DATA PREAMBLE
002466  131556  000004  123621          SUB     #4,R16          %CALL   S.ST16  ; [U52EC2][ECO#2]ADJUST FOR WORDS IN SERDES/RESET
002467  010557  007006  133604  D.IOPR: ADD     #BUF.SD,R17\N,BAR %CALL S.LD12  ; R12=POINTER TO SDI CTRL BLK
002470  030552  007001  133604          ADD     #SDI.SL,R12,BAR %CALL   S.LD12  ; R12=HEADER COMPARE SEARCH LIMIT
002471  013740  007136  133615          MOV     #HDLMT,BAR      %CALL   S.ST12  ; SET HEADER SEARCH LIMIT
002472  034451  000011  000000          CLR     R11                             ; CLEAR R11
002473  010557  007002  123607          ADD     #BUF.BP,R17\N,BAR %CALL S.LDDB  ; DBAR=POINTER TO START OF SECTOR DATA
002474  010557  007001  137777          ADD     #BUF.ST,R17\N,BAR %RET          ; [16K]BAR=PTR TO BUF.ST
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.BFCK
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL CHECK THE BUFFER STATUS OF THE
                                ; CURRENT D.PROC BUFFER AND WILL LOAD DMREG3 AND DMREG4 WITH
                                ; THE HEADER AND DATA PREAMBLES.
                                ;
                                ; INPUTS:
                                ;       DBAR    POINTS TO THE ACTIVE SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       CONDITION CODE  - REFLECTS STATUS OF BIT #BFULL,BUFFER STATUS
                                ;       DMREG3          HEADER PREAMBLE
                                ;       DMREG4          DATA PREAMBLE
                                ;       R13             0                                                               ;[E121]
                                ;       R12             SUSP+BFSV                                                       ;[E121]
                                ;       R11             SDI Status word (SDI.ST+DBAR)                                   ;[E121]
                                ;       R17             Points to buffer control block (SDI.DB+DBAR)                    ;[E121]
                                ;       R16             Buffer Status word (BUF.ST+R17)                                 ;[E121]
                                ;;-

002475  010555  007007  123611  D.BFCK: ADD     #SDI.DB,DBAR\N,BAR %CALL S.LD17 ; R17=CURRENT DPROC BUFFER ADDR
002476  010557  007000  133610  D.BFCA: ADD     #BUF.NL,R17\N,BAR %CALL S.LD16  ; R16=BUFFER STATUS
002477  034453  000013  122502          CLR     R13             %CALL   D.GPRE  ; GO GET HEADER AND DATA PREAMBLES
002500  033752  000140  133572          MOV     #<SUSP+BFSV>,R12 %CALL  D.GMST  ; R12=BITS TO SET IF BUFFER NOT RDY
002501  114556  000100  137777          BIT     #BFULL,R16      %RET            ; TEST BFULL BIT/RETURN

                                ; ***   ROUTINE TO GET HEADER AND DATA PREAMBLES INTO DMREG3 AND DMREG4 ***
002502  010555  007070  133604  D.GPRE: ADD     #SDI.DP,DBAR\N,BAR %CALL S.LD12 ;[V05] R12=DATA (LO) & HEADER (HI) PREAMBLES
002503  013740  007004  000000          MOV     #DMREG4,BAR                     ; BAR=DM REG 4 (HOLDS DATA PREAMBLE)
002504  014552  003377  133575          AND     #LOBYT,R12\N,BUF %CALL  S.SWAB  ; STORE DATA PREAMBLE/SWAP BYTES OF R12
002505  013740  007003  010000          MOV     #DMREG3,BAR                     ; BAR=DM REG 3 (HOLDS HEADER PREAMBLE)
002506  034552  003377  137777          AND     #LOBYT,R12,BUF  %RET            ; STORE HEADER PREAMBLE/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.CPUL (CHECK PULSE)
                                ;       D.WSIG (WAIT FOR RECVR RDY & R/W RDY SIGNALS)
                                ;
                                ; FUNCTIONAL DESCRIPTION
                                ;       THIS ROUTINE WILL WAIT FOR THE REAL TIME DRIVE STATUS BITS PASSED
                                ; IN R11 TO GO UP AND THEN COME DOWN.
                                ;
                                ; INPUTS:
                                ;       R11 - DRIVE STATUS BITS TO CHECK (D.CPUL)
                                ;       R11 - TIMEOUT VALUE (D.WSIG)
                                ;
                                ; OUTPUTS:
                                ;       NONE (D.CPUL)
                                ;       CCODE = 0 ERROR IN RECVR RDY OR R/W RDY (D.WSIG)
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

002507  014511  000002  000000  D.CPUL: BIT     (RTDS),R11                      ; CHECK DRIVE STATE FOR R11 FLAGS
002510  014511  010002  102510          BIT     (RTDS),R11      %JZRO   .       ; WAIT FOR IT TO RISE
002511  014511  010002  002511          BIT     (RTDS),R11      %JNZRO  .       ; WAIT FOR IT TO FALL
002512  033711  000002  137777          MOV     (RTDS),R11      %RET            ; R11=DRIVE STATE/RETURN

002513  013712  010002  167777  D.WSIG: MOV\F   (RTDS),R12\N    %RZRO           ; [16K]PUT RTDS ON BUS/RETURN IF ZERO
002514                          ASSUME  DRDY,EQ,BIT00                           ; [16K]MAKE SURE RECVR RDY IS LSB
002514  031471  040011  142513          DEC\T   R11             %JNLSB  D.WSIG  ; [16K]IF NO RECVR RDY THEN LOOP
002515                          ASSUME  RWRDY,EQ,BIT15                          ; [16K]MAKE SURE R/W RDY IS MSB
002515  031451  030011  177777  D.WSGA: DEC\F   R11             %RMSB           ; [16K]IF R/W THEN RETURN
002516  013712  010002  012515          MOV     (RTDS),R12\N    %JNZRO  D.WSGA  ; [16K]IF R11 NEQ 0 THEN LOOP
002517  034451  000011  127777          CLR     R11             %RET            ; [16K]CCODE = 0/RETURN
                                .PAGE                                                                   ;[E121]
```



```
                                ;;+                                                                     ;[E121]
                                ; ROUTINE NAME:                                                         ;[E121]
                                ;       D.WSAF  (WAIT FOR SAFE TIME TO ISSUE LEVEL 1 COMMAND)           ;[E121]
                                ;                                                                       ;[E121]
                                ; FUNCTIONAL DESCRIPTION:                                               ;[E121]
                                ;       D.WSAF waits until the drive has Receiver Ready and Read/Write Ready    ;[E121]
                                ;              asserted and is a comfortable distance from the next falling     ;[E121]
                                ;              edge of Sector/Index Pulse. It is equivalent to D.WSIG           ;[E121]
                                ;              followed by D.CPUL, but it is faster than that combination       ;[E121]
                                ;              under certain circumstances encountered while spiralling.        ;[E121]
                                ;                                                                               ;[E121]
                                ; INPUTS:                                                                       ;[E121]
                                ;       R16     Count controlling how long to wait for Rcvr Rdy & R/W Rdy       ;[E121]
                                ;                                                                               ;[E121]
                                ; OUTPUTS:                                                                      ;[E121]
                                ;       None - R11, R12, R14, R16 destroyed                                     ;[E121]
                                ;                                                                               ;[E121]
                                ;;-                                                                            ;[E121]
                                                                                                                ;[E121]
002520                          D.WSAF: ASSUME  RWRDY,EQ,BIT15                                                  ;[E121]
002520  053754  000003  010000          MOV\R   #DRDY*2+1,R14                   ; Set up mask for Rcvr Rdy & R/W Rdy    ;[E121]
002521  033751  000060  112533          MOV     #SEC+IDX,R11    %JMP    40$     ; Set up mask for Sector/Index Pulse    ;[E121]
                                                                                ; and enter state machine               ;[E121]
                                                                                                                        ;[E121]
                                ;       This routine is divided into three states:                              ;[E121]
                                ;                                                                               ;[E121]
                                ;       State 1 - Sector/Index low, position within sector unknown. If the      ;[E121]
                                ;       drive is ready to transfer here we must explicitly wait for a falling   ;[E121]
                                ;       edge of sector/index pulse.                                             ;[E121]
                                                                                                                ;[E121]
002522  031456  010016  102507  10$:     DEC     R16             %JZRO   D.CPUL  ; If drive ready, wait for edge   ;[E121]
002523  014511  010002  102507          BIT     (RTDS),R11      %JZRO   D.CPUL  ; If count expired, wait for edge   ;[E121]
002524  015514  010002  102522          BIC     (RTDS),R14\N    %JZRO   10$     ; Stay in state 1 until sector/index rises;[E121]
                                                                                                                ;[E121]
                                ;       State 2 - Sector/Index high. If the drive is ready to transfer here     ;[E121]
                                ;       we must explicitly wait for a falling edge of sector/index pulse.       ;[E121]
                                                                                                                ;[E121]
002525  031456  010016  102507  20$:     DEC     R16             %JZRO   D.CPUL  ; If drive ready, wait for edge   ;[E121]
002526  014511  010002  102507          BIT     (RTDS),R11      %JZRO   D.CPUL  ; If count expired, wait for edge   ;[E121]
002527  015514  010002  002525          BIC     (RTDS),R14\N    %JNZRO  20$     ; Stay in state 2 until sector/index falls;[E122D]
                                                                                                                ;[E121]
                                ;       State 3 - Sector/Index low, position known to be near start of sector.  ;[E121]
                                ;       If drive becomes ready to transfer here we can proceed immediately.     ;[E121]
                                                                                                                ;[E121]
002530  031456  010016  137777  30$:     DEC     R16             %RZRO           ; If drive ready, go for it!    ;[E121]
002531  031452  010012  102507          DEC     R12             %JZRO   D.CPUL  ; If count expired, wait for edge   ;[E121]
002532  015514  010002  012530          BIC     (RTDS),R14\N    %JNZRO  30$     ; Stay in state 2 for "Safe time"       ;[E121]
                                                                                                                ;[E121]
002533  033752  000264  010000  40$:     MOV     #180.,R12                       ; Safe time is about 190 uS, so         ;[E121]
                                                                                ; Safe loop ct is 190/(3*.346)          ;[E121]
002534  015514  000002  112522          BIC     (RTDS),R14\N    %JMP    10$     ; Enter loop in 1tate 1                 ;[E121]
                                                                                                                        ;[E121]
                                .PAGE                                                                           ;[E121]
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       D.SGRP  (SELECT GROUP)
                                ;       D.GRPC  (GROUP CLEANUP)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       D.SGRP ROUTINE WILL ISSUE THE SELECT GROUP REAL TIME COMMAND.
                                ;       D.GRPC ROUTINE WILL CLEANUP AFTER THE SELECT GROUP COMMAND.
                                ;
                                ; INPUTS:
                                ;       R17             POINTER TO BUFFER CONTROL AREA
                                ;
                                ; OUTPUTS:
                                ;       R12 IS DESTROYED
                                ;       R11 IS ZEROED
                                ;
                                ;;-

002535  013440  000000  132544  D.SGRP: NOP                             %CALL   D.GGRP  ; GO GET/RESET GROUP NUMBER
002536  133552  000216  101212          BIS     #SGRPCD,R12     %JMP    LEVOWR  ; OR IN SELECT GROUP OP CODE/SEND REAL TIME CMD/RET

002537  133752  000002  010000  D.GRPC: MOV     #1000,R12                       ; Set up count for loop                 ;[E121]
002540                          ASSUME  RWRDY,EQ,BIT15                  ; [ECO#2]MAKE SURE R/W RDY IS MSB       ;[E121]
002540  013712  010002  112542  10$:    MOV     (RTDS),R12\N    %JZRO   20$     ; [ECO#2]WAIT FOR R/W RDY TO GO AWAY    ;[E121]
002541  031452  030012  112540          DEC     R12             %JMSB   10$                                     ;[E121]
002542  033712  000003  000000  20$:    MOV     (BUF),R12                       ; [ECO#2]R12=STATUS                     ;[E121]
002543  035552  003001  100701          BIC     #BGRUP,R12,BUF  %JMP    D.IDLE  ; [ECO#2]CLEAR SELECT GROUP FLAG/GO IDLE

002544  010557  007011  133604  D.GGRP: ADD     #BUF.GP,R17\N,BAR %CALL S.LD12  ; [BDA]R12=THIS BUFFER'S GROUP
002545  034552  000377  010000          AND     #LOBYT,R12                      ; ISOLATE GROUP
002546  010557  007006  123603          ADD     #BUF.SD,R17\N,BAR %CALL S.LD11  ; R11=SDI CNTRL BLK PTR
002547  030551  007013  103615          ADD     #SDI.GP,R11,BAR %JMP    S.ST12  ; UPDATE GROUP/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       D.WHDR (WRITE HEADER PREAMBLE,SYNC,4 COPIES OF HEADER)
                                ;       D.WDAT (WRITE DATA PREAMBLE,SYNC,R16 WORDS OF DATA)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       ROUTINE D.WHDR IS USED TO REWRITE THE HEADER AREA OF A SECTOR
                                ; AND INCLUDES THE WRITING OF THE HEADER PREAMBLE, HEADER SYNC, FOUR
                                ; COPIES OF THE HEADER, GAP, AND SPLICE WORDS. IT WILL FLOW INTO
                                ; ROUTINE D.WDAT AND PROVIDE CORRECT TIMING TO FINISH WRITING THE
                                ; SECTOR DATA. IT WILL BE USED BY THE FORMAT XFC AND THE ROUTINE THAT
                                ; HANDLES THE 'MSCP' REPLACE COMMAND.
                                ;       ROUTINE D.WDAT IS USED TO WRITE A SECTOR'S DATA AREA AND
                                ; INCLUDES THE WRITING OF THE DATA PREAMBLE, DATA SYNC, AND DATA.
                                ; IT WILL BE USED BY THE D.WHDR AND D.WRT ROUTINES.
                                ;
                                ; INPUTS:
                                ;       D.WHDR          WRITE GATE SET
                                ;                       @SCMD MODE
                                ;                       R11 = 0
                                ;                       R12 = NUMBER OF WORDS IN HEADER PREAMBLE
                                ;                       R13 = LO DESIRED HEADER
                                ;                       R14 = HI DESIRED HEADER
                                ;
                                ;       D.WDAT          WRITE GATE SET
                                ;                       ECC RESET (@RECC)
                                ;                       DMREG4 = # WORDS IN DATA PREAMBLE
                                ;                       R16 = # WORDS TO TRANSFER
                                ;                       DBAR = POINTER TO DATA BUFFER
                                ;
                                ; OUTPUTS:
                                ;       D.WHDR,D.WDAT   ECC TIMING SET AND DATA AS DESCRIBED ABOVE
                                ;
                                ;;-
                                ; ***   WRITE HEADER PREAMBLE - R12 WORDS OF ZEROES ***                          ;[E121]
002550  021352  106011  052550  D.WHDR: DECB\F   R11\0,R12,SD    %JNDCLK D.WHDR  ; [16K]WHEN DRV CLK,ZAP SERDES/DECR R12
002551  013740  017202  042550          MOV\F    #SYNC,BAR       %JNZRO  D.WHDR  ; [ECO#2]DO R12 TIMES/POINT AT SYNC WORD
                                ; ***   WRITE HEADER SYNC ***
002552  013700  106003  052552  D.WHDA: MOV\F    (BUF),SD        %JNDCLK D.WHDA  ; OUTPUT HEADER SYNC
                                ; ***   WRITE HEADER (R13,R14) 4 TIMES ***
002553                          ASSUME  DMREG4,EQ,4                     ; MAKE SURE DMREG4=4
002553  033751  007004  010000          MOV     #DMREG4,R11,BAR                 ; R11=4/BAR=PTR TO DATA PREAMBLE SIZE
                                                                        ; *** PLEASE NOTE THAT DMREG4=4 AND
                                                                        ; R11 (HEADER LOOP COUNT) IS 4
002554  013440  106013  052554  D.WHDB: MOV\F    R13,SD          %JNDCLK D.WHDB  ; OUTPUT LO HEADER WORD
002555  021351  106014  052555  D.WHDC: DECB\F   R14\0,R11,SD    %JNDCLK D.WHDC  ; OUTPUT HI HEADER WORD
002556  033753  010002  052554          MOV\F    #2,R13          %JNZRO  D.WHDB  ; [EERREC][16K]WRITE 4 COPIES
                                ; ***   WRITE GAP,SPLICE, AND DATA PREAMBLE USING WORD COUNT IN R11 ***
002557  014440  106012  052557          CLR\F    R12,SD          %JNDCLK .       ; [16K]WHEN DRV CLK, SD=0
002560  034452  106012  052560          CLR\F    R12,R12,SD      %JNDCLK .       ; [16K]WHEN DRV CLK, SD=0
002561  033711  100503  042561          MOV\F    (BUF),R11 @RSE  %JNDCLK .       ; [16K]WHEN DRV CLK,RESET SERDES
002562  131551  000004  010000          SUB      #4,R11                         ; [16K]ADJUST DATA PREAMBLE
002563  031453  010653  012563          DEC      R13     @RCMD   %JNZRO  .       ; [16K]WAIT 6 CYCLES,RESET CMD MODE
```

```
KDBDP           DIGITAL EQUIPMENT CORP., 2901 ASSEMBLER VERSION 32              PAGE  178
D.PROC WRITE ROUTINE (*** NEW SDI TIMING ***)

002564  013740  007260  000000          MOV     #REPSTA,BAR                     ; [EERREC]GET REPLACE STATUS AREA
002565  013711  000002  000000          MOV     (RTDS),R11\N                    ; [EERREC]GET REAL TIME DRIVE STATE
002566                                  ASSUME  RWRDY,EQ,BIT15                  ; [EERREC]MAKE SURE READ/WRITE READY IS BIT 15
002566  013760  033001  052567          MOV\T   #1,BUF          %TNMSB          ; [EERREC]IF RWRDY NOT UP THEN FLAG AS A POTENTIAL PROB
002567  013440  000700  010000          NOP             @SSE                    ; [16K]ENABLE SERDES
                        ; ***   ENTRY POINT FOR WRITE ROUTINE ***   '
002570  021351  106652  052570  D.WDAT: DECB\F  R12\0,R11,SD @RCMD %JNDCLK D.WDAT; WHEN DRV CLK,SD=0/DECR R11/RESET COMMAND MODE
002571  013740  017202  052570          MOV\F   #SYNC,BAR       %JNZRO  D.WDAT  ; DO IT R11 TIMES
                        ; ***   WRITE DATA SYNC ***
002572  013700  106743  042572  D.WDTB: MOV\F   (BUF),SD @SECC  %JNDCLK D.WDTB  ; OUTPUT DATA SYNC
                        ; ***   OUTPUT DATA BUFFER (256 WORDS) AND EDC (1 WORD) ***
002573  033453  007015  000000          MOV     DBAR,R13,BAR                    ; [16K]R13,BAR=BUFFER ADDRESS
002574  031316  106003  042574  D.WDTC: DECB\F  (BUF),R16,SD    %JNDCLK D.WDTC  ; WHEN DRV CLK OUTPUT DATA  WORD
002575  130455  017015  012574          INC     DBAR,DBAR,BAR   %JNZRO  D.WDTC  ; INCR DBAR/LOOP 257 TIMES
                        ; ***   LET SERDES OUTPUT 17 ECC RESIDUES AND SET FOR NEXT SECTOR ***
                        ; ***   17 ECC SYMBOLS ARE OUTPUT IN 11 DCLKS FROM START OF @SECCT ***
002576  014440  106012  052576  D.WDTD: CLR\F   R12,SD          %JNDCLK D.WDTD  ; SD=ECC#1(258)
002577  013440  000000  127777          NOP                     %RET            ; [16K]RETURN
                                        .PAGE
```



```
KDBDP           DIGITAL EQUIPMENT CORP., 2901 ASSEMBLER VERSION 32              PAGE  179
D.PROC READ ROUTINE (*** NEW SDI TIMING ***)

                                        .SBTTL  D.PROC READ ROUTINE (*** NEW SDI TIMING ***)
                                ;;+     02-MAY-83       UDA52 MICROCODE
                                ; ROUTINE NAME:
                                ;       D.READ
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL READ DATA FROM THE DISK INTO BUFFERS ASSIGNED
                                ; TO THE SDI INTERCONNECT CONTROL BLOCK.
                                ;
                                ; INPUTS:
                                ;       DBAR            POINTER TO CURRENT SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       BUFFERS FILLED WITH DATA FROM DISK
                                ;       ALL D.PROC REGISTERS ARE USED
                                ;
                                ;;-

002600  013440  000000  132475  D.READ: NOP                             %CALL   D.BFCK  ; GO CHECK BUFFER STATUS
002601  013451  010611  012310          TST     R11     @RDPF   %JNZRO  D.NOBF  ; MAKE SURE PKT ACTIVE
                        ; ***   ALERT U.PROC FOR POTENTIAL FULL BUFFERS ***
002602                                  ASSUME  PKIP,EQ,BITOO                   ; MAKE SURE PKIP IS LSB
002602  033551  043100  152305  D.READA: BIS\F  #BFSV,R11,BUF   %JNLSB  D.IOEX  ; REQUEST BUFFER SERVICE
002603  010555  007006  133603          ADD     #SDI.UB,DBAR\N,BAR %CALL S.LD11 ; [U52EC1]R11=U.PROC BUFFER PTR
002604  133544  010200  110701          BIS     #PLOCK,RLL      %JZRO   D.IDLE  ; [U52EC1]IF EQ 0 THEN SKIP READ/UNLOCK U.PROC
002605  037156  000016  122520          XNOR    R16,R16         %CALL   D.WSAF  ; Wait for it to be Safe to xfer        ;[E121]
                        ; ***   FIRST ACQUIRE A BUFFER IF NEEDED [16K] ***
002606  033751  000002  122513  D.REDX: MOV     #2,R11          %CALL   D.WSIG  ; [ECO#2]GO WAIT FOR DRIVE SIGNALS
002607  010557  017002  102333          ADD     #BUF.BP,R17\N,BAR %JZRO D.RRWE  ; [16K]IF ERROR THEN GO FIND OUT WHICH ONE
002610  013440  000000  123006          NOP                     %CALL   D.ALBF  ; [U52EC2]GO ATTEMPT BUFFER ALLOCATION
002611  030577  010002  152670          ADD\T   #BUF.BP,R17     %JZRO   D.RNBF  ; [U52EC2]IF NO BUFFER THEN GET OUT     ;[E121]
002612  013740  005001  122467          MOV     #CONT,RTCS      %CALL   D.IOPR  ; [U52EC2]GO PREPARE FOR I/O
002613  033714  000523  010000  D.REDB: MOV     (BUF),R14 @SWM                  ; [16K]R14=BUF.ST/SET WRITE MODE
002614                                  ASSUME  BGRUP,EQ,BITOO                  ; [16K]MAKE SURE BGRUP IS LSB
002614  010577  047005  142620          ADD\T   #BUF.TA,R17\N,BAR %JNLSB D.REDC ; [16K]IF NO GROUP SELECT THEN CONTINUE
002615  013440  000000  132535          NOP                     %CALL   D.SGRP  ; [16K]GO ISSUE SELECT GROUP CMD
002616  010557  017001  012335          ADD     #BUF.ST,R17\N,BAR %JNZRO D.RRDE ; [16K]IF NEQ 0 THEN SELECT GROUP ERROR
002617  013712  000002  112537          MOV     (RTDS),R12\N    %JMP    D.GRPC  ; [16K]DO GROUP CLEANUP
                        ; ***   R12 = REAL TIME COMMAND - GO DO HEADER COMPARE ***
002620  033712  000443  123014  D.REDC: MOV     (BUF),R12 @SCMD %CALL   HDRCMP  ; [16K]R12=RTC,SET CMD MODE,GO DO HEADER COMPARE
002621  030577  017002  152633          ADD\T   #BUF.BP,R17,BAR %JZRO   D.REDE  ; [16K]IF MATCH THEN CONTINUE (2)        ;[E121]
                                                                                ; ** NOTE R17 IS MODIFIED HERE!         ;[E121]
002622  013440  000500  133064          NOP             @RSE    %CALL   HDRANL  ; RESET SERDES ENABLE/GO DO HEADER ANALYSIS;[E121]
002623  033711  010522  042352          MOV\F   (RTDS),R11 @SWM %JNZRO  D.HCER  ; [ECO#2]IF HEADER COMPARE ERROR THEN GO DISPATCH
                                                                                ; [ECO#2]R11=DRIVE STATE
002624                                  ASSUME  RWRDY,EQ,BIT15                  ; MAKE SURE RWRDY IS MSB
002624  114544  030100  002342          BIT     #DMODE,RLL      %JNMSB  D.LRER  ; [EERREC]IF NOT HI THEN ERROR/IF IN DM MODE
002625  050144  010004  052632          ADD\RF  RLL,RLL         %JNZRO  D.RED1  ; IF U.PROC LOCKING
002626  010557  027006  042632          ADD\F   #BUF.SD,R17\N,BAR %JNCRY D.RED1 ;   THEN CONTINUE
002627  033715  000003  010000          MOV     (BUF),DBAR                      ;   ELSE DBAR=SDI CONTROL BLK PTR
002630  010555  007000  123604          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD12 ; R12=SDI STATUS
002631  033552  003100  123570          BIS     #BFSV,R12,BUF   %CALL   UNLOCK  ; RESET SDI STATUS/UNLOCK U.PROC
002632  010557  007005  112620  D.RED1: ADD     #BUF.TA,R17\N,BAR %JMP  D.REDC  ; UNLOCK U.PROC/CONTINUE
```

```
                                ; ***   DESIRED HEADER FOUND - SET UP FOR DATA SYNC ***                                          ;[E121]
                                ;       New code added 1/15/87 to allow two shots at buffer lock - RL                            ;[E121]
                                                                                                                                 ;[E121]
002633  033715  000503  010000  D.REDE: MOV     (BUF),DBAR      @RSE            ; see if buf exists, kill SERDES                ;[E121]
002634  025564  017001  142637          BIC\T    #1,RLL\0,BAR    %JZRO   1$      ; br if none, & try for lock                    ;[E121]
002635  013440  000000  000000          NOP                      %JNCRY  D.DTER  ; buffer already in place - stall               ;[E121]
002636  013440  007017  112641          MOV      R17,BAR         %JMP    2$      ; and join "got buffer" code                    ;[E121]
002637  033715  040003  152653  1$:     MOV\F    (BUF),DBAR      %JNLSB  D.RLF1  ; br if failed, else get buf from list          ;[E121]
002640  021344  017017  152670          DECB\F   R17\0,RLL,BAR   %JZRO   D.RNBF  ; br if no buffers, else release lock          ;[E121]
                                                                                 ; and set up to store buf in ctl blk            ;[E121]
002641  013740  005001  010000  2$:     MOV      #CONT,RTCS                      ; clear frame sent flag                         ;[E121]
002642  033751  115005  052642          MOV\F    #<CONT+RD>,R11,RTCS    %JNCSR . ; wait for 1st flag, set RG & clear flag;[E121]
002643                                  ASSUME   BUF.BP,EQ,2                                                                     ;[E121]
002643  021357  115011  052643          DECB\F   R11\0,R17,RTCS  %JNCSR  .       ; wait for 2d flg, start restoring R17          ;[E121]
002644  013440  003015  010000          MOV      DBAR,BUF                        ; Store buffer in ctl blk                       ;[E121]
002645  021357  115011  052645          DECB\F   R11\0,R17,RTCS  %JNCSR  .       ; wait for 3d flg, finish restoring R17         ;[E121]
002646  033751  000340  000000          MOV      #340,R11                        ; Set R11 for data sync timeout                 ;[E121]
002647  013740  117002  042647          MOV\F    #DMREG2,BAR     %JNCSR  .       ; wait for 4th flg, get adr of sect lngth;[E121]
002650  033716  000003  010000          MOV      (BUF),R16                       ; R16 = sector length                           ;[E121]
002651  010557  007006  123605          ADD      #BUF.SD,R17\N,BAR %CALL S.LD13  ; R13 = SDI ctl blk ptr                         ;[E121]
002652  021356  007755  102673          DECB     DBAR\0,R16,BAR @SECC %JMP D.REDF; BAR=bufadr, predecr R16, set ECC mode         ;[E121]
                                                                                 ; and enter transfer code                       ;[E121]
002653  013740  007002  000000  D.RLF1: MOV      #DMREG2,BAR                     ; here on lock fail - get adr of sect ln;[E121]
002654  013740  005001  010000          MOV      #CONT,RTCS                      ; clear "frame sent" flg                        ;[E121]
002655  033751  115005  052655          MOV\F    #<CONT+RD>,R11,RTCS    %JNCSR . ; wait for 1st flg, set RG, clear flg          ;[E121]
002656  013440  115011  042656          MOV\F    R11,RTCS        %JNCSR  .       ; wait for 2d flg & clear it                    ;[E121]
002657  033716  000003  010000          MOV      (BUF),R16                       ; R16 = sector length                           ;[E121]
002660  021356  115011  052660          DECB\F   R11\0,R16,RTCS  %JNCSR  .       ; wait for 3d flg, predecr R16                  ;[E121]
002661  033751  000340  000000          MOV      #340,R11                        ; R11 = data sync timeout ctr                   ;[E121]
002662  025544  117001  052662          BIC\F    #1,RLL\0,BAR    %JNCSR  .       ; wait for 4th flg, try for lock again          ;[E121]
002663  033715  040003  152671          MOV\F    (BUF),DBAR      %JNLSB  D.RLF2  ; br if failed (again!) else get buf            ;[E121]
002664  021344  017017  152670          DECB\F   R17\0,RLL,BAR   %JZRO   D.RNBF  ; br if no bufs left else restore lock          ;[E121]
                                                                                 ; and set up to store buf in ctl blk            ;[E121]
002665                                  ASSUME   BUF.BP,EQ,2                                                                     ;[E121]
002665  021357  003015  010000          DECB     DBAR\0,R17,BUF                  ; Store buf in ctl blk, half-restore R17;[E121]
002666  010557  007005  123605          ADD      #BUF.SD-1,R17\N,BAR %CALL S.LD13; R13 = ptr to SDI ctl blk                      ;[E121]
002667  021357  007755  112673          DECB     DBAR\0,R17,BAR @SECC %JMP D.REDF; BAR=bufadr, restore R17, set ECC mode         ;[E121]
                                                                                 ; and enter data transfer code                  ;[E121]
002670  130444  000004  000000  D.RNBF: INC      RLL,RLL                         ; No buffers - restore lock to old value;[E121]
002671  131557  000002  123200  D.RLF2: SUB      #BUF.BP,R17     %CALL   D.ARTC  ; Two-time loser - restore R17, abort read;[E121]
002672  013440  000000  110701          NOP                      %JMP    D.IDLE  ; go find something else to do                  ;[E121]


                                ;       ********************************************************
                                ;       * The timing and subtlety of the follwoing code deserves*
                                ;       * clarification.                                       *
                                ;       * At D.REDF: a 1 instruction loop awaits               *
                                ;       * either the SERDES WRC going low or NCRY being asserted*
                                ;       * as these two signals are compined into the testmux   *
                                ;       * WRC condition.  The reception of the SYNC is indicated*
                                ;       * by SERDES WRC going low 7.5 bits after the SYNC is   *
                                ;       * received, for a duration of 8.5 bits.                *
                                ;       * Therefore the SYNC detection will be                 *
                                ;       * guaranteed at transfer rates upto 24.4 Mbits/sec.    *

KDBDP   KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```



```
                                ;       *                                                     *
                                ;       * The code following that guarantees that the SERDES   *
                                ;       * will be read no less than every two instructions, so  *
                                ;       * this guarantees reception at transfer rates up to:   *
                                ;       *       348ns*2/16bits = 43.5ns/bit = 22.9Mbits/sec   *
                                ;       *                                                     *
                                ;       ********************************************************


002673  031331  170706  052673  D.REDF: DECB\T   (SD),R11 @SSE   %JNWRC  D.REDF  ; DECR R11/ENABLE SERDES/WAIT FOR WORD RATE CLK
                                ; ***   PLEASE NOTE THAT THE @SECC AND @SSE IOCS MUST BE ON CONSECUTIVE CYCLES ***
                                ; ***   DATA SYNC FOUND ***
002674  031336  103006  152700          DECB\T   (SD),R16,BUF    %JDCLK  D.REDH  ; IF DRV CLK THEN DATA TO BUF,DEC R16
002675  031336  103006  152700          DECB\T   (SD),R16,BUF    %JDCLK  D.REDH  ; IF DRV CLK THEN DATA TO BUF,DEC R16
002676  013440  020000  002341          NOP                      %JNCRY  D.DTER  ; IF TIMEOUT THEN ERROR EXIT
002677  031316  103006  042677  D.REDG: DECB\F   (SD),R16,BUF    %JNDCLK D.REDG  ; IF NOT DRV CLK THEN WAIT/ELSE RD DATA/DECR R16
002700  130455  017015  012677  D.REDH: INC      DBAR,DBAR,BAR   %JNZRO  D.REDG  ; INCR DBAR,BAR/INPUT WORDS 1-255.
                                ; ***   IMPORTANT - SET ECC TIMING (@SECCT) WHEN LOADING DATA WORD 256. ***
002701  130315  103766  052701  D.REDI: INCB\F   (SD),DBAR,BUF @SECCT %JNDCLK D.REDI; INPUT WORD 256.,INCR DBAR,SET ECC TIMING
002702  013440  007015  000000          MOV      DBAR,BAR                        ; BAR=PTR TO EDC AREA
002703  013700  103006  052703          MOV\F    (SD),BUF        %JNDCLK .       ; INPUT WORD 257. (EDC)
002704  010557  007000  000000          ADD      #BUF.NL,R17\N,BAR               ; BAR=PTR TO NEXT BUFFER STATUS
002705  013711  100006  052705          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 258.
002706  033456  000017  000000          MOV      R17,R16                         ; SAVE R17 IN R16
002707  013711  100006  042707          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 259.
002710  033717  000003  000000          MOV      (BUF),R17                       ; R17=PTR TO NEXT LINK/STATUS THIS BUFFER
002711  013711  100006  052711          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 260.
002712  010557  007000  000000          ADD      #BUF.NL,R17\N,BAR               ; BAR=PTR TO NEXT LINK'S STATUS
002713  013711  100006  042713          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 261.
002714  033712  000003  000000          MOV      (BUF),R12                       ; R12=NEXT BUFFER'S STATUS
002715  013711  100006  042715          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 262.
002716  036152  000017  010000          XOR      R17,R12                         ; CHECK FOR SINGLE BUFFER RING
002717  013312  100006  042717          TSTB\F   (SD),R12        %JNDCLK .       ; INPUT WORD 263.
002720  037732  010003  042721          COM\T    (BUF),R12       %TNZRO          ; IF SINGLE BUFFER THEN FORCE R12=177777(BFULL ON)
002721  037312  100006  042721          COMB\F   (SD),R12        %JNDCLK .       ; INPUT WORD 264.
002722  010553  007001  000000          ADD      #SDI.SL,R13\N,BAR               ; [16K]BAR=PTR TO HEADER SEARCH LIMIT
002723  013711  100006  042723          MOV\F    (SD),R11\N      %JNDCLK .       ; INPUT WORD 265.
002724  033714  000003  000000          MOV      (BUF),R14                       ; [16K]R14=SEARCH LIMIT
                                ; ***   IMPORTANT - RESET ECC TIMING WHEN LOADING WORD 266. ***
002725  013711  100566  042725          MOV\F    (SD),R11\N @RECCT %JNDCLK .     ; INPUT WORD 266.,RESET ECC TIMING
002726  013740  007136  000000          MOV      #HDLMT,BAR                      ; [16K]BAR=PTR TO HEADER LIMIT
002727  034711  100006  052727          CLR\F    (SD),R11        %JNDCLK .       ; [16K]INPUT WORD 267.,CLR R11
002730  013440  003014  000000          MOV      R14,BUF                         ; [16K]STORE HEADER LIMIT
002731  037311  100006  052731          COMB\F   (SD),R11        %JNDCLK .       ; [16K]INPUT WORD 268.,CMPL R11
002732  013740  165001  112323          MOV      #CONT,RTCS      %JLATE  D.OVER  ; [ECO#2]DROP RD GATE/IF SERDES OVERRUN THEN ERROR
002733  010556  147000  112327          ADD      #BUF.NL,R16\N,BAR %JDSER D.RTDE ; [ECO#2]IF PARITY ERROR THEN PROCESS
002734  013712  000002  000000          MOV      (RTDS),R12\N                    ; [ECO#2]PRINE RR WAIT LOOP
002735                                  ASSUME   DRDY,EQ,BIT00                   ; [ECO#2]MAKE SURE RECVR RDY IS LSB
002735  013712  040002  102735          MOV      (RTDS),R12\N    %JNLSB  .       ; [ECO#2]WAIT FOR RR IN LINE FOR SPEED
002736                                  ASSUME   RWRDY,EQ,BIT15                  ; [ECO#2]MAKE SURE R/W RDY IS MSB
002736  013711  030507  012342          MOV      (ECC),R11\N @RSE %JNMSB D.LRER  ; [EERREC][16K][ECO#2]IF RW RDY UP THEN CONTINUE
002737  113577  033100  152764          BIS\T    #BFULL,R17\N,BUF %JMSB  D.REDJ  ; IF NO ECC ERROR THEN INDICATE BUFFER FULL/CONTINUE
                                ; ***   ECC ERROR RESIDUE PROCESSING (NOT REAL TIME) ***
002740  013740  007010  133615          MOV      #HEADER,BAR     %CALL   S.ST12  ; SAVE R12

KDBDP   KDB50.MICROCODE.,22-APR-1988 11:27:16.96
```

```
002741  114544  000100  010000            BIT     #DMODE,RLL                          ; [U52EC1]IF NOT IN DM MODE
002742  010553  017024  133623            ADD     #SDI.OE,R13\N,BAR %CZRO S.CLRB      ; [U52EC1]  THEN DISABLE OVERLAP POSSIBILITY
002743  033757  000006  010000            MOV     #6,R17                              ; R17=LOOP COUNT
002744  013440  000000  133002  D.RECC:   NOP                         %CALL   D.GECC  ; GET ECC RESIDUE
002745  033452  000014  123002            MOV     R14,R12             %CALL   D.GECC  ; GET ECC RESIDUE
002746  033751  000012  133574            MOV     #10.,R11            %CALL   S.ROTL  ; ADJUST 1ST RESIDUE BITS 0-5 LEFT 10 BITS
002747  033451  000012  010000            MOV     R12,R11                             ; SAVE RESULT IN R11
002750  134551  000374  123005            AND     #176000,R11         %CALL   D.SETB  ; ISOLATE UPPER 6 BITS/SET BAR
002751  033151  003014  123002            OR      R14,R11,BUF         %CALL   D.GECC  ; PACK 1ST ENTRY/GET ECC RESIDUE
002752  033751  000012  133574            MOV     #10.,R11            %CALL   S.ROTL  ; ADJUST 1ST RESIDUE BITS 6-9 LEFT 10 BITS
002753  134552  000374  123005            AND     #176000,R12         %CALL   D.SETB  ; ISOLATE UPPER 6 BITS/SET BAR
002754  031457  000017  000000            DEC     R17                                 ; DECREMENT LOOP COUNTER
002755  033152  013014  002744            OR      R14,R12,BUF         %JNZRO  D.RECC  ; PACK 2ND ENTRY/IF NOT ZERO THEN LOOP
                                  ; ***   RESIDUES PACKED, SET UP TO CONTINUE ***
002756  033751  000060  132507            MOV     #<SEC*IDX>,R11      %CALL   D.CPUL  ; [ECO#2]RESYNCH TO SECTOR PULSE
002757  013740  007010  133604            MOV     #HEADER,BAR         %CALL   S.LD12  ; RESTORE R12
002760  010556  007001  123611            ADD     #BUF.ST,R16\N,BAR %CALL S.LD17      ; [16K]R17=BUF.ST
002761  133557  003020  000000            BIS     #BECC,R17,BUF                       ; [16K]INDICATE ECC ERROR
002762  010556  007000  133611            ADD     #BUF.NL,R16\N,BAR %CALL S.LD17      ; R17=STATUS & NEXT LINK
002763  113557  003100  010000            BIS     #BFULL,R17\N,BUF                    ; [16K]SET FULL
                                  ; ***   UPDATE D.PROC DATA BUFFER POINTER ***
002764                            .        ASSUME  BLAST,EQ,BIT15                      ; MAKE SURE BLAST IS MSB
002764  010553  037007  102776  D.REDJ:   ADD     #SDI.DB,R13\N,BAR %JMSB D.REDN      ; IF LAST BUFFER THEN DON'T RESET SDI.DB
002765  013440  003537  010000            MOV     R17,BUF @SWM                        ; [ECO#2]ELSE RESET SDI.DB/SET WRITE MODE
002766  050144  000544  000000            ADD\R   RLL,RLL @RECC                       ; [ECO#2]ATTEMP TO LOCK OUT U.PROC/RESET ECC
002767  010553  027000  012773            ADD     #SDI.ST,R13\N,BAR %JNCRY D.REDL     ; IF LOCKED THEN CONTINUE
002770  033711  000003  000000            MOV     (BUF),R11                           ;   ELSE R11=SDI STATUS
002771                            .        ASSUME  PKIP,EQ,BIT00                       ; MAKE SURE PKIP IS LSB
002771  033571  043100  042772            BIS\T   #BFSV,R11,BUF       %TLSB           ; IF PKT ACTIVE THEN SET BUFFER SERVICE FLAG
002772  133544  000200  010000            BIS     #PLOCK,RLL                          ; FREE U.PROC
002773  114552  000100  000000  D.REDL:   BIT     #BFULL,R12                          ; IF THIS BUFFER EMPTY
002774  010557  017001  112613            ADD     #BUF.ST,R17\N,BAR %JZRO D.REDB      ;   [16K]THEN GO GET NEXT SECTOR
002775  033454  000532  102231  D.REDM:   MOV     R12,R14 @RRM        %JMP    D.IODN  ;   ELSE DO I/O DONE PROCESSING
002776  010553  007024  133606  D.REDN:   ADD     #SDI.OE,R13\N,BAR %CALL S.LD14      ; [U52EC1]R14=OVERLAP ENABLE FLAG
002777  114544  000100  010000            BIT     #DMODE,RLL                          ; [U52EC1]IF NOT IN DMODE
003000  132454  010014  123617            NEG     R14,R14             %CZRO   S.ST14  ; [U52EC1]  THEN NEGATE AND RESTORE
003001  133556  000200  112775            BIS     #BLAST,R16          %JMP    D.REDM  ; SET BLAST IN R16/FINISH UP

                                  ; ***   ECC RESIDUE PROCESSING ROUTINES ***
003002  033714  000627  000000  D.GECC:   MOV     (ECC),R14 @SRSGEN                   ; R14=ECC RESIDUE/SET RSGEN
003003  013440  000420  000000            NOP             @RRSGEN                     ; THANKS CURT !/RESET RSGEN
003004  135554  000374  137777            BIC     #176000,R14         %RET            ; ISOLATE 10 BITS OF RESIDUE

003005  130455  007015  137777  D.SETB:   INC     DBAR,DBAR,BAR       %RET            ; SET BAR, INCR DBAR, RETURN

                                  ; ***   D.PROC DATA BUFFER ALLOCATION ROUTINE ***
                                  ; ***   ENTER WITH BAR -> BUF.BP
                                  ; ***   RETURN CCODE = ZERO IF NO BUFFER AVAILABLE, = NZERO IF BUFFER OBTAINED OR EXISTED
                                  ; ***   NOTE !!! - IF RETURN CCODE = ZERO THEN RLL IS STILL LOCKED !!!!!!!!
003006  033716  000003  010000  D.ALBF:   MOV     (BUF),R16                           ; [U52EC2][16K]R16=BUFFER ADDRESS
003007  013440  010000  077777            NOP\F                       %RNZRO          ; [U52EC2][16K]IF GOT IT THEN CONTINUE
003010  025544  007001  000000            BIC     #1,RLL\0,BAR                        ; [U52EC2][16K]IF BUFFER LOCK CLEAR
003011  033716  040003  153010            MOV\F   (BUF),R16           %JNLSB  .-1     ; [U52EC2][16K]  THEN WAIT FOR IT
003012  031444  010004  167777            DEC\F   RLL,RLL             %RZRO           ; [U52EC2][16K]IF NO BUFFER THEN EXIT
```



```
003013  010557  007002  103621            ADD     #BUF.BP,R17\N,BAR %JMP  S.ST16   ; [U52EC2][16K]SAVE BUFFER POINTER
                                  .PAGE
```

```
                                .SBTTL  HEADER COMPARE AND ANALYSIS ROUTINES
                              ; ***    NEW SDI TIMING ***
                              ;;+      09-JULY-83       UDA52 MICROCODE
                              ; ROUTINE NAME:
                              ;        HDRCMP
                              ;
                              ; FUNCTIONAL DESCRIPTION:
                              ;        THIS ROUTINE PERFORMS THE HEADER COMPARE FUNCTION FOR THE UDA.
                              ; THIS IS THE PRIMARY MEANS OF ROTATIONAL POSITION SENSING FOR THE UDA.
                              ; THE HEADER IS 32 BITS LONG AND IS REPLICATED 4 TIMES. THE HEADER COMPARE
                              ; ALGORITHM USED IN THE UDA IS AS FOLLOWS:
                              ;        THE HEADER IS TO BE BROKEN INTO TWO 16 BIT FIELDS (LOW AND HIGH).
                              ; IF ANY THREE OF THE FOUR LOW FIELDS AS STORED ON THE DISK MATCHES THE LOW
                              ; FIELD DESIRED, AND ANY TWO OF THE FOUR HIGH FIELDS AS STORED ON THE DISK
                              ; MATCHES THE HIGH FIELD DESIRED, THE HEADER COMPARE SUCCEEDS.
                              ;
                              ; INPUTS:
                              ;        WRITE MODE SET (@SWM)
                              ;        COMMAND MODE SET (@SCMD)
                              ;        R12 = REAL TIME COMMAND
                              ;        R17 = POINTER TO CURRENT SECTOR BUFFER
                              ;        THE ISSUING OF THE LEVEL 0 COMMAND IS SYNCHRONIZED WITH SECTOR/INDEX PULSE
                              ;
                              ; OUTPUTS:
                              ;        R13 AND R14 CONTAIN THE DESIRED HEADER
                              ;        R11 IS AN ERROR INDICATOR - ON RETURN R11=0 MEANS SUCCESS,
                              ;        R11>0 MEANS DRIVE CLOCK TIMEOUT, R11<0 MEANS HEADER TIMEOUT
                              ;        R12 IS A PARTIAL MATCH INDICATOR - ONCE UPDATED BASED ON THE RETURNED
                              ;        CONDITION CODES (SEE BELOW), IF R12 & 210 = 0 THE HEADER READ FROM
                              ;        THE DISK IS THE DESIRED ONE, OTHERWISE IT IS NOT
                              ;
                              ;        IF THE CONDITION CODES ON RETURN ARE ZERO THEN R12 SHOULD BE DECREMENTED BY 20
                              ;        TO YIELD THE FINAL MATCH INDICATOR, OTHERWISE R12 IS THE FINAL MATCH INDICATOR.
                              ;
                              ;        WHENEVER R11 IS NON-ZERO UPON RETURN, R12 IS FORCED TO 177777.
                              ;
                              ;;-

003014  013740  007202  133055  HDRCMP: MOV     #SYNC,BAR       %CALL   D.SRTC  ; [16K]BAR=PTR TO SYNC/SEND RTC
003015  014512  100502  053015          BIT\F   (RTDS),R12 @RSE %JNDCLK .       ; [16K]WHILE PUTTING SERDES IN READ MODE
003016  014512  010642  103016          BIT     (RTDS),R12 @RCMD %JZRO  .       ; [16K]WAIT FOR SECTOR/INDEX PULSE
003017  014512  010722  013017          BIT     (RTDS),R12 @SRM %JNZRO  .       ; [16K]WAIT FOR SECTOR/INDEX TRAILING EDGE
003020  170456  000711  133053          INC\L   R11,R16 @SSE    %CALL   D.LDHD  ; ENABLE SERDES, GET HEADER IN (R14,R13)
003021                                  ASSUME  WRT,EQ,2                        ; R16=2 FOR WRITE GATE SET AT D.WRTC+1
003021  013740  007010  000000          MOV     #HEADER,BAR                     ; [ECO#2]BAR=HI HEADER SAVE AREA
003022  037752  170145  113024          COM     #145,R12        %JWRC   HDRCME  ; [US2EC2][ECO#2]R12=232(LO BYTE),ALL ONES(HI BYTE)
                              ; ***    THE PREVIOUS COM SETS UP TWO FOUR BIT COUNTERS IN THE LOW BYTE OF R12. THE HIGH 4 BITS
                              ; ***    COUNTS THE HIGH HEADER MATCHES AND THE LOW 4 BITS COUNTS THE LOW HEADER MATCHES.
                              ; ***    A MATCH IS MADE WHEN THE COUNTERS ARE DECREMENTED TO BE LESS THAN 8. THEREFORE
                              ; ***    EACH 4 BIT COUNTER IS INITIALLY SET TO BE 7 PLUS THE NUMBER OF MATCHES DESIRED TO SATISFY
                              ; ***    THE MATCH CRITERIA. CURRENTLY HI COUNTER = 9. AND LO COUNTER = 10.

003023  031572  170377  043023  HDRCMD: SUBC\T  #377,R12        %JNWRC  HDRCMD  ; [ECO#2]WAIT FOR WORD RATE CLOCK
003024  016533  103006  153030  HDRCME: XOR\T   (SD),R13\N,BUF  %JDCLK  LO1A    ; [ECO#2]MATCH R13 TO 1ST LO FIELD/TIMEOUT
```



```
003025  016533  103006  153030          XOR\T   (SD),R13\N,BUF  %JDCLK  LO1A    ; [ECO#2]MATCH R13 TO 1ST LO FIELD/TIMEOUT

003026  013440  020000  013050          NOP                     %JNCRY  HDRCMF  ; [ECO#2]INIT HI/LO FIELD MATCH INDICATOR
003027  016513  103006  053027  LO1:    XOR\F   (SD),R13\N,BUF  %JNDCLK .       ; MATCH R13 TO 1ST LO FIELD/TIMEOUT
003030  031472  010012  153031  LO1A:   DEC\T   R12,R12          %TZRO           ; [ECO#2]IF MATCH THEN MASSAGE INDICATOR
003031  016514  103006  053031  HI1:    XOR\F   (SD),R14\N,BUF  %JNDCLK .       ; MATCH R14 TO 1ST HI FIELD/TIMEOUT
003032  131572  010020  153033          SUB\T   #20,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003033  016513  103006  053033  LO2:    XOR\F   (SD),R13\N,BUF  %JNDCLK .       ; MATCH R13 TO 2ND LO FIELD/TIMEOUT
003034  031472  010012  143035          DEC\T   R12,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003035  016514  103006  043035  HI2:    XOR\F   (SD),R14\N,BUF  %JNDCLK .       ; MATCH R14 TO 2ND HI FIELD/TIMEOUT
003036  131572  010020  143037          SUB\T   #20,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003037  016513  103006  043037  LO3:    XOR\F   (SD),R13\N,BUF  %JNDCLK .       ; MATCH R13 TO 3RD LO FIELD/TIMEOUT
003040  031472  010012  143041          DEC\T   R12,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003041  016514  103006  043041  HI3:    XOR\F   (SD),R14\N,BUF  %JNDCLK .       ; MATCH R14 TO 3RD HI FIELD/TIMEOUT
003042  131572  010020  143043          SUB\T   #20,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003043  016513  103006  043043  LO4:    XOR\F   (SD),R13\N,BUF  %JNDCLK .       ; MATCH R13 TO 4TH LO FIELD/TIMEOUT
003044  031472  010012  153045          DEC\T   R12,R12          %TZRO           ; IF MATCH THEN MASSAGE INDICATOR
003045  016514  103006  053045  HI4:    XOR\F   (SD),R14\N,BUF  %JNDCLK .       ; [16K]MATCH R14 TO 4TH HI FIELD/TIMEOUT
003046  131572  010020  153047          SUB\T   #20,R12          %TZRO           ; [16K]IF MATCH THEN MASSAGE INDICATOR
003047  034552  000210  137777          AND     #210,R12         %RET            ; [16K]CHECK FOR 2 LO/2 HI/RETURN

                              ; ***    WORD RATE CLOCK TIMEOUT ERROR RECOVERY PROCEDURE ***
003050  013440  000500  131270  HDRCMF: NOP             @RSE    %CALL   D.CLCS  ; RESET SERDES ENABLE/CLEAR CTLR STATE BITS
003051  037151  000531  010000          XNOR    R11,R11 @SWM                    ; SET WRITE MODE
003052  037152  000452  127777  HDRCMG: XNOR    R12,R12 @SCMD   %RET            ; SET COMMAND MODE/RETURN

                              ; ***    LOAD HEADER'S INTO R13,R14 ***
003053  010557  007003  123605  D.LDHD: ADD     #BUF.HL,R17\N,BAR %CALL S.LD13  ; R13=LO HEADER
003054  010557  007004  103606          ADD     #BUF.HH,R17\N,BAR %JMP  S.LD14  ; R14=HI HEADER/RETURN

                              ; ***    SEND REAL TIME COMMAND FOR READ,WRITE,FORMAT OPERATION ***
003055  013440  070700  003055  D.SRTC: NOP             @SSE    %JACLO  .       ; [16K]ENABLE SERDES
003056  013700  106003  053056          MOV\F   (BUF),SD         %JNDCLK .       ; [16K]OUTPUT SYNC TO SERDES
003057  013440  106012  053057          MOV\F   R12,SD           %JNDCLK .       ; [16K]OUTPUT REAL TIME COMMAND
003060  014440  106011  053060          CLR\F   R11,SD           %JNDCLK .       ; [16K]CLEAR SERDES
003061  013740  007003  010000          MOV     #DMREG3,BAR                     ; [16K]BAR=PTR TO DMREG3 FOR FORMAT
003062  034451  106011  053062          CLR\F   R11,R11,SD       %JNDCLK .       ; [16K]CLEAR SERDES AND R11
003063  033752  000060  137777          MOV     #<IDX+SEC>,R12   %RET            ; [16K]RETURN
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       HDRANL
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM HEADER MATCH FAILURE ANALYSIS.
                                ; IF THE HEADER COMPARE FAILS AND THERE ARE AT LEAST TWO OUT OF FOUR
                                ; LOW FIELD MATCHES THEN THE STORED HIGH FIELDS ARE EXAMINED FOR A CORRECT
                                ; POSITION AND DIFFERING HEADER CODES (I.E. REVECTOR TO THIS TRACK,OR
                                ; REVECTOR). THIS ANALYSIS IS PERFORMED DURING THE SECTOR TIME OF THE
                                ; MATCH FAILING SECTOR. IF THE LOW ORDER HEADER WORDS DIDN'T MATCH, THE
                                ; HEADER IS CHECKED FOR INTERNAL CONSISTENCY AND THEN COMPARED AGAINST THE
                                ; DESIRED HEADER TO DETERMINE IF WE ARE ON THE RIGHT TRACK. IF WE ARE, THEN
                                ; THE HEADER SEARCH LIMIT IS DECREMENTED.
                                ;
                                ; INPUTS:
                                ;       R11             ERROR INDICATOR FROM HDRCMP ROUTINE
                                ;       R12             HEADER MATCH INDICATOR FROM HDRCMP ROUTINE
                                ;       R13,R14         DESIRED HEADER FROM HDRCMP ROUTINE
                                ;       R17             BUFFER POINTER
                                ;       DBAR            POINTS TO DATA PORTION OF BUFFER
                                ;       FLAG            SET IF PRIMARY AND SECONDARY REVECTORING IS TO BE IGNORED (USED BY REPLACE)
                                ;
                                ; OUTPUTS:
                                ;       R12,R16 USED AS TEMPORARY REGISTERS
                                ;       R17 PRESERVED; DBAR PRESERVED IF R11 = 0 ON RETURN
                                ;       R11 IS THE OUTPUT STATUS:
                                ;               R11=-1          ON TRACK BUT VERY FAR AWAY-RESCHEDULE
                                ;               R11=0           NO ERRORS, KEEP LOOKING
                                ;               R11=1           [ECO#2]NOT USED
                                ;               R11=2           FAILURE IN DRIVE - RETRY SEQUENCE NECESSARY
                                ;               R11=3           SECONDARY REVECTORING REQUIRED
                                ;               R11=4           SEARCH LIMIT EXCEEDED - TERTIARY REVECTORING REQUIRED
                                ;               R11=5           SPECIAL RETURN FOR REPLACE - ON TRACK NO ERRORS
                                ;               R11=6           HEADER COMPARE ERROR ON XBN
                                ;
                                ;       IN THE CASE OF PRIMARY OR TERTIARY REVECTORING, BUF.HL/HH ARE REPLACED
                                ;       BY THE HEADER OF THE 1ST REPLACEMENT BLOCK ON THIS TRACK AND THE SEARCH
                                ;       LIMIT IS RE-INITIALIZED; THE DESIRED LBN IS SAVED IN (OLDLBN,OLDLBN+1).
                                ;
                                ;;-

003064  033456  000014  133200  HDRANL: MOV     R14,R16         %CALL   D.ARTC  ; [16K]SAVE HI HEADER/GO ABORT R/T COMMAND
003065  013451  000011  000000          TST     R11                             ; CHECK ERROR STATE WORD
003066  013440  010000  113103          NOP                     %JZRO   HDRANA  ; [16K]BRANCH IF NO ERRORS
003067  033751  000017  132513          MOV     #15.,R11        %CALL   D.WSIG  ; [16K]GO WAIT FOR RCVR RDY & R/W RDY
003070  033771  010002  177777          MOV\T   #2,R11          %RZRO           ; [16K]RETURN HARD ERROR IF SOMETHING IS WRONG
                                                                                ; ELSE TREAT AS A MISSED HEADER

                                ;       COME HERE IF EVERYTHING LOOKS OK BUT THE HEADER DOESN'T MATCH (OR DIDN'T READ)

003071  010557  007002  123607  HDRNOM: ADD     #BUF.BP,R17\N,BAR %CALL S.LDDB  ; DATA BUFFER POINTER
003072  013740  007136  123603          MOV     #HDLMT,BAR      %CALL   S.LD11  ; GET SEARCH COUNT
003073  031471  010011  063614          DEC\T   R11,R11         %CNZRO  S.ST11  ; [U52EC2]DECREMENT IT AND STORE IT BACK
```



```
003074  034451  010011  027777          CLR     R11             %RNZRO          ; RETURN OK IF LIMIT NOT REACHED
003075  010557  007004  133603          ADD     #BUF.HH,R17\N,BAR %CALL S.LD11  ; GET HIGH DESIRED HEADER WORD
003076  134551  000360  000000          AND     #HDCOD,R11                      ; ISOLATE HEADER TYPE
003077  033751  010004  137777          MOV     #4,R11          %RZRO           ; IF LBN SET TERTIARY REVECT FLAG AND RETURN
003100  114544  000100  010000          BIT     #DMODE,RLL                      ; [16K]IF IN DM MODE
003101  013451  010011  027777          TST     R11             %RNZRO          ;   THEN SET TERTIARY REVECT
003102  033751  000006  137777          MOV     #6,R11          %RET            ;   ELSE MUST BE FCT OR RBN HEADER

                                ;       COME HERE TO ANALYZE A HEADER WE READ

003103  033454  000016  000000  HDRANA: MOV     R16,R14                         ; RESTORE HI HEADER
003104  014552  000010  010000          BIT     #10,R12                         ; DID LOW-ORDER MATCH?
003105  114554  010360  013121          BIT     #HDCOD,R14      %JNZRO  HDRANB  ; IF SO, CHECK DESIRED HEADER TYPE
003106  136554  010120  013121          XOR     #RBNPRM,R14     %JNZRO  HDRANB  ; IF NOT LBN, FORGET REVECTORING
003107  033751  060035  023246          MOV     #35,R11         %CNDPF  HDRCKH  ; CHECK FOR PRIMARY REVECT (IF ENABLED)
003110  136554  010140  113210          XOR     #RBNXOR,R14     %JZRO   RBNPRI  ; IF PRIMARY, GO DO IT NOW...
003111  033751  060035  023246          MOV     #35,R11         %CNDPF  HDRCKH  ; NOW CHECK FOR 2DARY REVECT (IF ENABLED)
003112  033771  010003  167777          MOV\T   #3,R11          %RZRO           ; IF SO, RETURN SPECIAL CODE.
003113  136554  000240  010000          XOR     #H11XOR,R14                     ; [U52EC2]R14=BAD HEADER CODE
003114  033751  060035  023246          MOV     #35,R11         %CNDPF  HDRCKH  ; [U52EC2]NOW CHECK FOR BAD HDR CODE
003115  013440  010000  013071          NOP                     %JNZRO HDRNOM   ; [U52EC2]IF NO MATCH GO AWAY
003116  013740  007136  133623          MOV     #HDLMT,BAR      %CALL   S.CLRB  ; [U52EC2] HDR CODE 11 FAKES NOT FOUND
003117  010557  007001  133603          ADD     #BUF.ST,R17\N,BAR %CALL S.LD11  ; [U52EC2] BUT SETS "ON-TRACK" FLAG
003120  133551  003010  113071          BIS     #BGOOD,R11,BUF  %JMP    HDRNOM  ; [U52EC2]

                                ;       NOT REVECTOR HEADER - CHECK THAT WE ARE ON THE RIGHT TRACK...
                                ;       FIRST CHECK THE DISK HEADER FOR INTERNAL CONSISTENCY, THEN (IF ITS CONSISTENT)
                                ;       CHECK IF IT IS THE SAME TYPE AS THE DESIRED HEADER AND IF IT DIFFERS
                                ;       BY MORE THAN THE MAX NUMBER OF THAT TYPE HEADER ON A TRACK.
                                ;       IF IT DOES, SOMETHING HAS GONE AWRY.......

003121  013740  007010  123243  HDRANB: MOV     #HEADER,BAR     %CALL   HDRCCL  ; SEE IF 1ST LO HEADER APPEARS 3 TIMES
003122  033473  010014  143125          MOV\T   R14,R13         %JZRO   HDRANC  ; IF SO, SAVE AS DISK LO HEADER
003123  013740  007012  133243          MOV     #HEADER+2,BAR   %CALL   HDRCCL  ; SEE IF 2D LO HEADER APPEARS 3 TIMES
003124  033453  010014  043071          MOV\F   R14,R13         %JNZRO  HDRNOM  ; IF NOT, INCONSISTENT HDR - CAN'T USE IT.
003125  013740  007011  133245  HDRANC: MOV     #HEADER+1,BAR   %CALL   HDRCCH  ; SEE IF 1ST HI HEADER APPEARS 3 TIMES
003126  033476  010014  143131          MOV\T   R14,R16         %JZRO   HDRAND  ; IF SO, WE HAVE A CONSISTANT HEADER
003127  013740  007013  123245          MOV     #HEADER+3,BAR   %CALL   HDRCCH  ; SEE IF 2D HI HEADER APPEARS 3 TIMES
003130  033456  010014  043071          MOV\F   R14,R16         %JNZRO  HDRNOM  ; IF NOT, INCONSISTANT HEADER
                                ; ***   R13 = CONSISTANT LO HEADER, R14,R16 = CONSISTANT HI HEADER ***
003131  010557  007004  133603  HDRAND: ADD     #BUF.HH,R17\N,BAR %CALL S.LD11  ; GET DESIRED HI HEADER
003132  036156  000011  000000          XOR     R11,R16                         ; [EC0#2]SEE IF HEADER TYPE OF DISK AND DESIRED
003133  114551  000360  010000          BIT     #HDCOD,R11                      ; [U52EC2]IF DESIRED HDR IS NOT LBN
003134  134556  010360  003137          AND     #HDCOD,R16      %JNZRO  HDRAD1  ; [U52EC2]  THEN CONTINUE/ISOLATE XOR'ED HDR CODE
003135  136576  010120  043136          XOR\T   #RBNPRM,R16     %TNZRO          ; If neq 0 then test for prim revect    ;[E121]
003136  136576  010060  053137          XOR\T   #RBN2ND,R16     %TNZRO          ; If neq 0 then test for 2dary revect   ;[E121]
003137  013440  010000  013071  HDRAD1: NOP                     %JNZRO  HDRNOM  ; [V05]IF NOT, DON'T USE
003140  114544  000100  133204          BIT     #DMODE,RLL      %CALL   HDSUBA  ; [V05]R16 -> SUBUNIT CHAR
003141                          ASSUME  LBNMSK,EQ,LOBYT                         ; INSURE MASK IS LO BYTE
003141  010556  007074  133612          ADD     #SDI.12,R16\N,BAR %CALL S.LL12  ; GET LBN'S/TRACK IN R12
003142  134551  000360  000000          AND     #HDCOD,R11                      ; ISOLATE HDR TYPE - IS HEADER AN LBN?
003143  010556  017065  103150          ADD     #SDI.GC,R16\N,BAR %JZRO HDRANE  ; IF SO, POINT AT START LBN & GO DIVIDE
003144  010556  007067  133604          ADD     #SDI.RT,R16\N,BAR %CALL S.LD12  ; GET RBN'S/TRACK IN R12
003145  034552  000077  010000          AND     #RBNMSK,R12                     ; ISOLATE RBN'S/TRACK
```

```
003146  136551  000140  010000          XOR     #RBNCOD,R11                     ; COMPARE HEADER TYPE AGAINST RBN
003147  010556  017066  003071          ADD     #SDI.TG,R16\N,BAR %JNZRO HDRNOM ; IF NOT RBN, FORGET IT
003150  033456  000013  133254  HDRANE: MOV     R13,R16         %CALL   D.DBNO  ; DIVIDE HEADER TO GET SECTOR OFFSET
003151  132151  000016  123053          RSUB    R16,R11         %CALL   D.LDHD  ; NORMALIZE LO HEADER TO START OF TRACK,
                                                                                ; GET DESIRED HEADER IN (R14,R13)
003152  131153  000011  000000          SUB     R11,R13                         ; FIND OFFSET FROM BEGINNING OF FOUND TRACK
003153  010557  007001  133606          ADD     #BUF.ST,R17\N,BAR %CALL S.LD14  ; [16K][ECO#2]R14=POSITIONER GOODNESS FLAG
003154  131153  000012  000000          SUB     R12,R13                         ; SEE IF OFFSET IS LEGAL (< TRACK SIZE)
003155  114544  020100  113071          BIT     #DMODE,RLL      %JCRY   HDRNOM  ; [16K][ECO#2]IF WRONG TRK THEN DECR HDRLMT/IF IN DM MODE
003156  133554  013010  013071          BIS     #BGOOD,R14,BUF  %JNZRO  HDRNOM  ;   [ECO#2]THEN TREAT AS ORDINARY MISS
003157  033771  060005  177777          MOV\T   #5,R11          %RDPF           ; SPECIAL RETURN FOR REPLACE CMD
                                                                                                                ;[E121]
                                ; ***   EVERYTHING OK - BUT LET'S SEE HOW FAR WE ARE AWAY FROM TARGET ***        ;[E121]
                                                                                                                ;[E121]
003160  010557  007006  123605          ADD     #BUF.SD,R17\N,BAR %CALL S.LD13  ; Get SDI ctl blk ptr from buf          ;[E121]
003161  010553  007067  123605          ADD     #SDI.RT,R13\N,BAR %CALL S.LD13  ; Get RBN's/track from unit char        ;[E121]
003162  034553  000077  000000          AND     #RBNMSK,R13                     ; Mask out crap                         ;[E121]
003163  030152  000013  000000          ADD     R13,R12                         ; Add to LBN's/tk to get sects/tk       ;[E121]
003164  010557  007003  123605          ADD     #BUF.HL,R17\N,BAR %CALL S.LD13  ; Reload desired lo header              ;[E121]
003165  131153  000016  010000          SUB     R16,R13                         ; Sub desired LBN from actual LBN       ;[E121]
003166  030173  030012  153167          ADD\T   R12,R13         %TNEG           ; If negative, wrap around              ;[E121]
003167  111553  000012  000000          SUB     #10.,R13\N                      ; If difference less then 10 sectors    ;[E121]
003170  031473  030013  153175          DEC\T   R13,R13         %JNEG   1$      ; then sit and wait for the right sector;[E121]
003171  010555  007023  123603          ADD     #SDI.RO,DBAR\N,BAR %CALL S.LD11 ; R11=ROTATIONAL OPTIM COUNT
003172  031471  010011  063614          DEC\T   R11,R11         %CNZRO  S.ST11  ; IF NEQ 0 THEN DECR RESET
003173  013455  010015  153071          TST\F   DBAR            %JZRO   HDRNOM  ; [ECO#2]IF ZERO THEN HANG IN THERE ANYWAY
003174  037151  000011  127777          XNOR    R11,R11         %RET            ;   ELSE R11:=-1/RETURN

003175  013440  010000  103071  1$:     NOP                     %JZRO   HDRNOM  ; [mjt07] if zero, normal exit/else r13 has #sectors away
003176  033751  000060  132507          MOV     #<IDX+SEC>,R11  %CALL   D.CPUL  ; [mjt07] wait for sector or index pulse
003177  031453  000013  113175          DEC     R13,R13         %JMP    1$      ; [mjt07] loop until on sector

                                ; ***   ROUTINE TO ABORT REAL TIME COMMAND ***
003200  013740  005005  121015  D.ARTC: MOV     #<CONT+RD>,RTCS %CALL   X.SSTA  ; [16K]SET READ GATE FOR A WHILE (32 BIT TIMES)
003201  013740  005005  121015          MOV     #<CONT+RD>,RTCS %CALL   X.SSTA  ; [16K]SO WE CAN LOWER IT AND ABORT THE I/O
003202  010557  007006  131006          ADD     #BUF.SD,R17\N,BAR %CALL X.SSDT  ; [16K]RELOAD DBAR WITH SDI CTL BLK PTR
003203  033715  000003  111006          MOV     (BUF),DBAR      %JMP    X.SSDT  ; [16K]WHILE WE LOWER READ GATE

                                ; *** ROUTINE TO SET R16 AS POINTER TO SUBUNIT CHAR [V05]
003204  010555  017002  153207  HDSUBA: ADD\F   #SDI.CP,DBAR\N,BAR %JZRO HDSLB1 ; [V05] IF NOT IN DM MODE,BRANCH
003205  033716  000003  010000          MOV     (BUF),R16                       ; [V05] ELSE, GET ADDRESS FROM DM PROGRAM
003206  131556  000056  137777          SUB     #SDI.CW,R16     %RTN            ; [V05] AND ADJUST/EXIT
003207  033456  000015  137777  HDSLB1: MOV     DBAR,R16        %RTN            ; [V05] IN OPERATIONAL MODE,SET R16=DBAR

                                .PAGE
```



```
                                ;       ROUTINE TO COMPUTE PRIMARY RBN FOR CURRENT CYL,GROUP,TRACK AND PLACE IT
                                ;       AS AN RBN HEADER (ALSO USABLE AS AN RCT OFFSET FOR TERTIARY REVECTORING)
                                ;       IN BUF.HH AND BUF.HL
                                ;       ON ENTRY, DBAR --> SDI CTL BLK, R17 --> BUFFER
                                ;       ON EXIT, R11 = BUF.HL(R17) = LO HEADER, R12 = BUF.HH(R17) = HI HEADER

                                ; ***   ENTRY FOR PRIMARY RBN'S, SAVE ORIG LBN AWAY FOR LATER USE ***
003210  135554  000360  133234  RBNPRI: BIC     #HDCOD,R14      %CALL   D.SLBA  ; [U52EC2]SAVE R13/R14 IN P.BUF2/3
003211  010555  007001  133604  RBNCLC: ADD     #SDI.SL,DBAR\N,BAR %CALL S.LD12 ; RESET HEADER LIMIT,
003212  013740  007136  133615          MOV     #HDLMT,BAR      %CALL   S.ST12  ; SINCE WE WILL BE CHANGING THE SEARCH TGT
003213  114544  000100  133204          BIT     #DMODE,RLL      %CALL   HDSUBA  ; GET R16 -> SUBUNIT CHAR
003214                                  ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE
003214  010556  007074  133612          ADD     #SDI.12,R16\N,BAR %CALL S.LL12  ; GET LBN'S/TRACK
003215  010556  007065  133254          ADD     #SDI.GC,R16\N,BAR %CALL D.DBNO  ; DIVIDE LBN BY LBN'S/TK TO GET TRACK #
003216  010556  007066  123604          ADD     #SDI.TG,R16\N,BAR %CALL S.LD12  ; GET WORD CONTAINING START RBN
003217  134552  000017  122015          AND     #RBNSTR,R12     %CALL   S.CL11  ; R12<11:8>=START RBN, CLEAR R11
003220  010556  007067  133610          ADD     #SDI.RT,R16\N,BAR %CALL S.LD16  ; GET RBN'S/TRACK IN R16
003221  034556  000077  000000          AND     #RBNMSK,R16                     ; ISOLATE RBN'S/TRACK
003222  030151  010013  153226  RBNCLA: ADD\F   R13,R11          %JZRO   RBNCLB  ; MULTIPLY LOOP - MULTIPLY THE ABSOLUTE
003223  130472  020012  153224          INC\T   R12             %TCRY           ; TRACK NUMBER IN (R14,R13) BY
003224  030152  000014  010000          ADD     R14,R12                         ; THE NUMBER OF RBN'S/TRACK IN R16
003225  031456  000016  103222          DEC     R16             %JMP    RBNCLA  ; TO GET THE FIRST RBN ON THIS TRACK.
003226  010557  007003  123614  RBNCLB: ADD     #BUF.HL,R17\N,BAR %CALL S.ST11  ; RESULT IS IN (R12,R11) STORE IT
003227  010557  007002  123607          ADD     #BUF.BP,R17\N,BAR %CALL S.LDDB  ; RESTORE DATA BUFFER PTR FOR PRIMARY CASE
003230  010557  007004  010000          ADD     #BUF.HH,R17\N,BAR               ; IN THE BUFFER AS THE NEW DESIRED HEADER
003231  133552  003140  010000          BIS     #RBNCOD,R12,BUF                 ; (SETTING THE RBN HEADER TYPE, OF COURSE)
003232  014451  000011  137777          CLR     R11,R11\N       %RET            ; RETURN CCODE ZERO FOR PRIMARY CASE

                                ; ***   ROUTINE TO SAVE R13/R14 IN SDI.RL AND SDI.RH - USES R12 ***
                                ; ***   CALL MUST DO "BIT    #HDCOD,R14", WILL NOT SAVE NON-LBN'S ***
003233  013440  010000  027777  D.SLBN: NOP                     %RNZRO          ; [U52EC2]IF NOT LBN THEN RETURN
003234  010555  007022  123604  D.SLBA: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD12 ; R12=PKT PTR
003235  114544  000100  010000          bit     #dmode,r11                      ; test 4 fix --test if dm
003236  013440  010000  003241          nop                             %jnzro 1$       ; skip
003237  010555  007031  133616          ADD     #SDI.RL,DBAR\N,BAR %CALL S.ST13 ; [QDA]SAVE LO LBN
003240  010555  007032  113617          ADD     #SDI.RH,DBAR\N,BAR %JMP S.ST14  ; [QDA]SAVE HI LBN/RETURN
003241  010552  007014  133616  1$:     add     #p.buf2,r12\n,bar   %call s.st13
003242  010552  007015  103617          add     #p.buf3,r12\n,bar   %jmp s.st14 ;
                                .PAGE
```

```
                                        ;;+
                                        ;       SUBROUTINES USED BY HDRANL:
                                        ;
                                        ;       HDRCCH,HDRCCL   CHECK CONSISTENCY OF HEADER HI/LO WORDS (LOOK FOR 3 COPIES OF (BUF))
                                        ;               INPUTS:         (BUF) = MATCH WORD
                                        ;               OUTPUTS:        CC = ZRO IF MATCH
                                        ;
                                        ;       HDRCKH,HDRCKL   CHECK EQUALITY OF HEADER HI/LO WORDS (LOOK FOR N COPIES OF R14)
                                        ;               INPUTS:         R14 = MATCH WORD, R11 = 33 + N
                                        ;               OUTPUTS:        CC = ZRO IF N COPIES OF R14 FOUND
                                        ;;-

003243  033751  000036  133606  HDRCCL: MOV     #36,R11         %CALL   S.LD14  ; SET UP R14 AND R11 FOR HDRCKL
003244  033756  000006  103247  HDRCKL: MOV     #<HEADER-2>,R16 %JMP    HDRCHK  ; SET UP BASE OF LO HDRS FOR HDRCHK
003245  033751  000036  133606  HDRCCH: MOV     #36,R11         %CALL   S.LD14  ; SET UP R14 AND R11 FOR HDRCKH
003246  033756  000007  010000  HDRCKH: MOV     #<HEADER-1>,R16                 ; SET UP BASE OF HI HDRS FOR HDRCHK
003247  030556  037002  113253  HDRCHK: ADD     #2,R16,BAR      %JMSB   HDRCKE  ; ADDRESS NEXT COPY OF HEADER, EXIT IF NONE
003250  016514  000003  000000          XOR     (BUF),R14\N                     ; COMPARE WITH DESIRED VALUE
003251  031471  010011  143252          DEC\T   R11,R11         %TZRO           ; COUNT IT IF EQUAL
003252  131551  000010  113247          SUB     #10,R11         %JMP    HDRCHK  ; DECREMENT HDR COUNT IN R11<5:3> & LOOP
003253  034551  000004  127777  HDRCKE: AND     #4,R11          %RET            ; DONE - RETURN Z BIT IF WE FOUND MATCHES

                                        ;       UNSIGNED DIVIDE ROUTINE
                                        ;       ENTER AT D.DBNO WITH (R14,R13) CONTAINING BLOCK NUMBER (LBN OR RBN),
                                        ;       R12 = DIVISOR, BAR --> STARTING BLOCK NUMBER<27:24> IN <11:8>
                                        ;       OR ENTER AT D.DIVD WITH (R14,R13) CONTAINING A 32-BIT NUMERATOR,
                                        ;       R12 = DIVISOR, R11 = 0.
                                        ;       IN EITHER CASE, ON EXIT, (R14,R13) = QUOTIENT, R11 = REMAINDER, R12 INTACT

003254  135554  000360  133603  D.DBNO: BIC     #HDCOD,R14      %CALL   S.LD11  ; GET START BLOCK NUM IN R11
003255                                  ASSUME  LBNSTR,EQ,RBNSTR
003255  134551  000017  000000          AND     #LBNSTR,R11                     ; ISOLATE STARTING BLOCK NUMBER
003256  131154  000011  132015          SUB     R11,R14         %CALL   S.CL11  ; MAKE BLOCK # UNIT-RELATIVE, CLEAR R11
003257  013740  007017  133621  D.DIVD: MOV     #<HEADER+7>,BAR %CALL   S.ST16  ; SAVE R16 SO WE CAN USE IT FOR COUNT
003260  033756  000041  000000          MOV     #33.,R16                        ; INITIALIZE ITERATION COUNT
003261  131151  000012  010000  D.DVDA: SUB     R12,R11                         ; TRIAL SUBTRACT TO TEST DIVISIBILITY
003262  030151  020012  143264  D.DVDB: ADD\F   R12,R11         %JCRY   D.DVDC  ; JUMP IF YES / RESTORE R11 IF NO
003263  030153  000013  113265          ADD     R13,R13         %JMP    D.DVDD  ; START DOUBLING NUMERATOR/QUOTIENT
003264  130153  000013  000000  D.DVDC: ADDC    R13,R13                         ; START DOUBLING BUT SET LO BIT TO 1
003265  031456  020016  113271  D.DVDD: DEC     R16             %JCRY   D.DVDF  ; DECR ITER CNT, CHECK CARRY TO HI NUM/QUO
003266  030154  010014  103272          ADD     R14,R14         %JZRO   D.DVDG  ; NO CARRY, DOUBLE HI NUM/QUO, CHK FOR END
003267  130151  020011  103261  D.DVDE: ADDC    R11,R11         %JCRY   D.DVDA  ; DOUBLE R11, ASSUME CARRY, LOOP IF CORRECT
003270  031151  000012  113262          SUBC    R12,R11         %JMP    D.DVDB  ; WRONG, ADJUST TRIAL SUBTRACT AND LOOP
003271  130154  010014  013267  D.DVDF: ADDC    R14,R14         %JNZRO  D.DVDE  ; R13 CARRIED, DBL R14 W/CARRY & CHK FOR END
003272  013740  007017  113610  D.DVDG: MOV     #<HEADER+7>,BAR %JMP    S.LD16  ; RESTORE R16 AND EXIT

                                        .PAGE
```



```
                                        .SBTTL  REVECTORING PROCESS
                                        ;;+     19-SEPT-1983    UDA/QDA MICROCODE
                                        ;
                                        ; REVECTORING CODE
                                        ;
                                        ; THIS CODE IS STRUCTURED AS A PROCESS WHICH RUNS SOMETIMES UNDER THE UNIBUS
                                        ; PROCESSOR AND SOMETIMES UNDER THE DISK PROCESSOR.  IT IS DISPATCHED TO FROM
                                        ; THE IDLE LOOPS OF THESE PROCESSORS.
                                        ;
                                        ; WHEN REVECTORING IS DESIRED FOR A DRIVE, THE "RVCT" BIT IS SET IN ITS SDI
                                        ; CONTROL BLOCK STATUS WORD (SDI.ST). WHEN THE REVECTORING CONTEXT BLOCK
                                        ; (RAM LOCATIONS REVSTR THROUGH REVEND) BECOMES FREE (REVSDI=0) THEN REVSDI
                                        ; IS SET TO POINT TO THE SDI NEEDING REVECTORING AND THIS PROCESS IS ENTERED
                                        ; AT U.RVC1 BY THE UPPER PROCESSOR. THE VECTORING IS UNDER CONTROL OF THE
                                        ; WORD RVCVEC, WHICH ACTS AS A "PC" POINTING TO THE NEXT SECTION TO BE EXECUTED;
                                        ; IF RVCVEC IS EVEN A DISK PROCESSOR CODE SECTION IS INVOKED, WHILE IF IT IS
                                        ; ODD A UNIBUS PROCESSOR SECTION IS INVOKED; IN EITHER CASE RVCVEC IS A POINTER
                                        ; TO A DISPATCH IN THE UDA VECTOR TABLE.
                                        ;
                                        ; INPUTS:
                                        ;               RVCBUF POINTS TO BUFFER NEEDING REVECTORING
                                        ;               UBAR/DBAR POINTS TO SDI CONTROL BLOCK
                                        ;
                                        ; OUTPUTS:
                                        ;               SDI.DB ADVANCED PAST BUFFER (IF APPROPRIATE),
                                        ;               OPERATION COMPLETE, DISK
                                        ;               ON CORRECT CYL/GROUP/TRACK FOR SUBSEQUENT OPERATIONS.
                                        ;;-
                                                .PAGE
```

```
                                ;       NEXT, THE LOWER PROCESSOR SAVES BUFFER PARAMETERS FROM THE BUFFER IN
                                ;       ERROR (@RVCBUF) AND COMPUTES THE INITIAL OFFSET INTO THE RCT
                                ;       FOR THE LBN NEEDING REVECTORING.
                                ;       THIS IS PURELY COMPUTATIONAL AND COULD BE DONE IN THE UPPER PROCESSOR AS WELL,
                                ;       BUT WE CAN USE THE "RBNCLC" ROUTINE IN HEADER ANALYZE TO DO SOME OF THE WORK...

                                D.RVC2::MOV     R17,BUF         %JMP    D.RVC2  ; ACTUALLY DONE IN THE DISPATCH VECTOR...
003273  010557  007000  133610        ADD     #BUF.NL,R17\N,BAR %CALL S.LD16  ; R16=STATUS & NEXT LINK OF REVECTOR CAUSING BUFFER
003274  113556  003200  000000        BIS     #BLAST,R16\N,BUF                ; INDICATE THAT ITS LAST BUFFER IN CHAIN
                                                                                ; FOR DURATION OF REVECTOR OPERATION
003275  013740  007230  123621        MOV     #RVCEBF,BAR     %CALL   S.ST16  ; SAVE BUF.NL OF BUFFER CAUSING REVECTOR
                                ; ***   SAVE THIS COMMAND'S PARAMETERS FROM MSCP PACKET (LATER SMASHED BY CONVERT ) ***
003276  033756  000005  123365        MOV     #<S.SECI-S.CYLL>,R16 %CALL D.RSAV ; R16:# WORDS TO MOVE/GO SET UP FOR MOVE
003277  033454  000012  010000        MOV     .R12,R14                        ; R14=R12
003300  033752  000123  132105        MOV     #RVCSAV,R12     %CALL   D.SMOV  ; R12=SAVE AREA/GO SAVE PARAMS
                                ; ***   SAVE ORIGINAL LBN AND BUF.TA ***
003301  010557  007005  133604        ADD     #BUF.TA,R17\N,BAR %CALL S.LD12  ; GET TRACK ADDRESS & OP CODE
003302  013740  007132  123615        MOV     #RVCTA,BAR      %CALL   S.ST12  ; SAVE ORIGINAL REAL TIME OP CODE
003303  034552  000377  133053        AND     #LOBYT,R12      %CALL   D.LDHD  ; ISOLATE TRACK ADDRESS
003304  013740  007130  133616        MOV     #RVCBNL,BAR     %CALL   S.ST13  ; STORE THE ORIGINAL HEADER FOR ERROR REPORTING
003305                                ASSUME  <RVCBNH&17>,EQ,<RVCBNL&17+1>
003305  133552  000027  133617        BIS     #READCD,R12     %CALL   S.ST14  ; REMOVE LEVEL 1 CODE TO READ (FOR RCT)
003306  010557  007005  133615        ADD     #BUF.TA,R17\N,BAR %CALL S.ST12  ; RESTORE IN BUFFER FOR RBNCLC
003307  013440  000000  123210        NOP                     %CALL   RBNPRI  ; [U52EC2]SAVE BUF.HL/H,SET UP BUF.SD,
                                                                                ; [U52EC2] CALCULATE INITIAL RCT OFFSET
003310  033453  000011  000000        MOV     R11,R13                         ; SAVE COPY OF LOW OFFSET IN R13
003311  074551  000177  010000        AND\L   #177,R11                        ; R11 = INITIAL WORD OFFSET WITHIN RCT BLOCK
003312  013740  007222  123614        MOV     #RCTOFF,BAR     %CALL   S.ST11  ; SAVE INITIAL OFFSET
003313  035553  000177  000000        BIC     #177,R13                        ; REMOVE INTRA-BLOCK PORTION OF OFFSET IN R13
003314  034552  000177  000000        AND     #177,R12                        ; REMOVE HIGH-ORDER CRUD FROM HI OFFSET WORD
003315  073152  000013  123575        BIS\L   R13,R12         %CALL   S.SWAB  ; COMBINE WORDS TOGETHER
                                                                                ; R12 LEFT 9 (RIGHT 7) = INITIAL BLOCK NUMBER
003316  013740  007224  133615        MOV     #RCTBLK,BAR     %CALL   S.ST12  ; [16K]SAVE INITIAL RCT BLOCK OFFSET (FROM BASE)
003317  013740  007252  123610  D.RV2A: MOV     #RVECDP,BAR     %CALL   S.LD16  ; [UQA]GET PROCESS VECTOR POINTER
003320  130456  000016  123621        INC     R16             %CALL   S.ST16  ; [UQA]INCREMENT IT AND STORE IT
003321  013740  007254  123610  D.RVDX: MOV     #RVCFLG,BAR     %CALL   S.LD16  ; [UQA]R16=U.PROC/D.PROC FLAG
003322  017440  003016  110701        COM     R16,BUF         %JMP    D.IDLE  ; [UQA]SWITCH TO U.PROC/GO IDLE
                                        .PAGE
```



```
                                ;       AT THIS POINT "SEEK" IS SET IN SDI.ST, WHICH TAKES PRECEDENCE OVER REVECTORING;
                                ;       WHEN THE SEEK COMPLETES THE LOWER PROCESSOR WILL COME HERE TO READ THE DESIRED
                                ;       RCT BLOCK INTO THE BUFFER POINTED TO BY RVCBUF

                                ;       THIS CODE IS USED BOTH TO READ THE RCT AND TO PERFORM THE FINAL READ/WRITE
                                ;       OF THE REVECTOR TARGET BLOCK

                                ;       AT THIS POINT, THE SEEK HAS BEEN ISSUED FOR THE REVECTOR BLOCK; WHEN IT IS DONE,
                                ;       WE WILL PERFORM THE USER'S ORIGINAL READ OR WRITE ON THAT BLOCK.

                                D.RVC6::MOV     (BUF),R17       %JMP    D.RVC4  ; ACTUALLY DONE IN DISPATCH VECTOR...

                                D.RVC4::MOV     (BUF),R17       %JMP    D.RVC4  ; ACTUALLY DONE IN DISPATCH VECTOR...
003323  014551  000004  010000        BIT     #DERR,R11                       ; SEE IF WE GOT A SEEK ERROR
003324  013440  010000  003337        NOP                     %JNZRO  D.RV4B  ; IF SO, SKIP THE READ AND REPORT AN ERROR
003325  010557  007005  133604        ADD     #BUF.TA,R17\N,BAR %CALL S.LD12  ; LOOK AT THE SDI LEVEL 1 COMMAND
003326  010557  007001  133603        ADD     #BUF.ST,R17\N,BAR %CALL S.LD11  ; [16K]R11=BUFFER STATUS
003327  135551  000020  000000        BIC     #BECC,R11                       ; [16K]CLEAR ECC FLAG
003330  035551  003001  010000        BIC     #BGRUP,R11,BUF                  ; [16K]CLEAR BGRUP FLAG IN BUF.ST
003331  010557  007000  123603        ADD     #BUF.NL,R17\N,BAR %CALL S.LD11  ; GET BUFFER STATUS/LINK WORD IN R11
003332  013452  000012  000000        TST     R12                             ; CHECK IF OPERATION IS A READ OR A WRITE
003333                                ASSUME  READCD,GT,0
003333                                ASSUME  WRITCD,LT,0
003333  133571  033100  142123        BIS\T   #BFULL,R11,BUF  %JMSB   D.WRIT  ; GO EXECUTE A WRITE (BUFFER FULL),
003334  135551  003100  111002        BIC     #BFULL,R11,BUF  %JMP    D.ERRD  ; [ECO#2]DO READ USING CURRENT R17

                                ; ***   D.IODN AND D.IOER RETURN HERE, CC=ZERO IF OK ***

003335  013440  010000  103317  D.RV4A: NOP                     %JZRO   D.RV2A  ; THIS STEP OVER NOW IF READ WORKED
003336  013740  007226  133611        MOV     #RVCBUF,BAR     %CALL   S.LD17  ; R17=PTR TO REVECTOR BUFFER
003337  010557  007001  133603  D.RV4B: ADD     #BUF.ST,R17\N,BAR %CALL S.LD11  ; [16K]READ FAILED - GET BUFFER STATUS WORD
003340  133551  000020  000000        BIS     #BECC,R11                       ; [16K]SET ECC FLAG
003341  033551  003001  103317        BIS     #BGRUP,R11,BUF  %JMP    D.RV2A  ; [16K]SET FATAL ERROR AND RETURN TO IDLE LOOP
                                        .PAGE
```

```
                                ;       FINAL PHASE - CLEAN UP AND FREE THIS SDI CTL BLK AND THE REVECTOR PROCESS
                                ;       NOTE - TO PREVENT RACE CONDITIONS THE REVECTOR PROCESS MUST BE FREED FROM THE SAME
                                ;       PROCESSOR THAT ALLOCATED IT (IN THIS CASE DPROC), OR THE RVCT BIT MUST BE CLEARED FIRST.

                                ; ***   RESTORE ORIGINAL PARAMETERS TO END OF MSCP PACKET ***
                                D.RVC8:;MOV     (BUF),R17       %JMP    D.RVC8  ; ACTUALLY DONE IN DISPATCH VECTOR
003342  033756  000005  123365          MOV     #<S.SECI-S.CYLL>,R16  %CALL D.RSAV ; R16=LOOP COUNTER/GO SET UP FOR RESTORE
003343  014551  000004  010000          BIT     #DERR,R11                       ; IF NOT FATAL ERROR
003344  033754  010123  122105          MOV     #RVCSAV,R14     %CZRO   D.SMOV  ;   THEN GO RESTORE I/O PARAMS
003345  033753  000160  010000          MOV     #<BFRQ+BFSV+SUSP>,R13           ; R13=SDI STATUS BITS TO CLR
                                                                                ; ASSUME ERROR - CLEAR BUF REQ+BUF SRV
                                ; ***   DO FINAL CLEANUP OF SDI STATUS, ETC. ***
003346  010557  007001  123604          ADD     #BUF.ST,R17\N,BAR %CALL S.LD12  ; [EERREC]GET BUFFER STATUS WORD
003347  014552  000040  010000          BIT     #BRCTS,R12                      ; [EERREC]CHECK IF THIS IS RCT SEARCHED CASE
003350  013440  010000  103353          NOP                     %JZRO   D.RV8A  ; [EERREC]IF NOT THEN SKIP
003351  010557  007000  133604          ADD     #BUF.NL,R17\N,BAR %CALL S.LD12  ; [EERREC]GET NEXT LINK PTR
003352  133552  003100  000000          BIS     #BFULL,R12,BUF                  ; [EERREC]SET BFULL TO FAKE UPPER INTO RECOVERY
003353  013740  007232  123623  D.RV8A: MOV     #RVCSDI,BAR     %CALL   S.CLRB  ; RELEASE THE REVECTOR PROCESS
003354  013740  007250  133623          MOV     #RVECUP,BAR     %CALL   S.CLRB  ; [U0A]CLEAR U.PROC VECTOR
003355  033752  000240  122306          MOV     #<XCMP+SUSP>,R12 %CALL  D.CLUR  ; [US2EC2]AND SET TRANSFER COMPLETE AND SUSPEND
003356  133553  000200  010000          BIS     #RVCT,R13                       ; [US2EC2]ALWAYS CLEAR RVCT, AS WE ARE DONE
003357  010555  007000  133603          ADD     #SDI.ST,DBAR\N,BAR %CALL S.LD11 ; [US2EC2]RELOAD R11 WITH SDI.ST
003360  014551  000004  010000          BIT     #DERR,R11                       ; CHECK FOR ERROR IN REVECT PROCESS OR FINAL SEEK
003361  133573  010100  042302          BIS\T   #ERRIP,R13      %JNZRO  D.SETS  ; [US2EC2]IF SO THEN CLEAR ERRIP/SET SUSP+XCMP
003362  013457  000017  000000          TST     R17                             ; AND SETTING SUSPEND IFF THE REVECTORED
003363                                  ASSUME  BLAST,EQ,BIT15
003363  033752  030100  002302          MOV     #BFSV,R12       %JNMSB  D.SETS  ; PREPARE TO EXIT BY SETTING BUFF SERVICE REQ,
003364  033552  000040  102302          BIS     #SUSP,R12       %JMP    D.SETS  ; BUFFER WAS LAST IN CHAIN/GO CHG SDI STATUS

                                ; ***   ROUTINE FOR SAVING AND RESTORING MSCP I/O PARAMETERS ***
003365  010555  007022  123604  D.RSAV: ADD     #SDI.PQ,DBAR\N,BAR %CALL S.LD12 ; R12=MSCP PACKET POINTER
003366  030552  000024  127777          ADD     #S.CYLL,R12     %RET            ; [US2EC2]ADD SAVE START OFFSET/RETURN
                                        .PAGE
```



```
                                ; ***   NEW SDI TIMING ***
                                        .SBTTL  D.PROC XFC ROUTINES
                                ;;+     22-JAN-83       UDA52 MICROCODE
                                ; ROUTINE NAME:
                                ;       XFC01
                                ;
                                ; FUNCTIONAL DESCRIPTION;
                                ;       THIS ROUTINE WILL PROCESS THE FORMAT TRACK ON INDEX XFC. THIS
                                ; COMMAND WILL ENABLE A DM PROGRAM TO FORMAT AN ENTIRE DISK. DM REGISTER
                                ; 0 POINTS TO A TABLE WHOSE ENTRIES CONTAIN A DATA BUFFER POINTER, LO ORDER
                                ; HEADER CODE, AND HI ORDER HEADER CODE. THE HEAD ADDRESS FOR THE FORMAT
                                ; COMMAND IS IN DM REGISTER 1 AND DM REGISTER 2 CONTAINS THE SECTOR SIZE IN
                                ; WORDS. THE FIRST SECTOR IS FORMATTED USING A FORMAT ON INDEX REAL TIME
                                ; COMMAND AND SUBSEQUENT SECTORS ARE FORMATTED USING THE FORMAT ON SECTOR
                                ; REAL TIME COMMAND. THE FORMAT OPERATION IS TERMINATED WHEN A TABLE ENTRY
                                ; CONTAINS A ZERO DATA BUFFER POINTER.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO FORMAT CONTROL TABLE
                                ;       DM REGISTER 1           HEAD ADDRESS
                                ;       DM REGISTER 2           SECTOR SIZE IN WORDS
                                ;       DM REGISTER 3           HEADER PREAMBLE SIZE IN WORDS
                                ;       DM REGISTER 4           DATA PREAMBLE SIZE IN WORDS
                                ;       DM REGISTER 5           POINTER TO RECIRCULATE LIST ENTRY
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 0           POINTS AT LAST DATA BUFFER PROCESSED
                                ;       DM REGISTER 1           0 - SUCCESSFUL COMPLETION
                                ;                               1 - DRIVE SIGNAL TIMEOUT
                                ;                               2 - READ/WRITE READY NOT HIGH
                                ;
                                ;;-

                                ; ***   FORMAT INTIALIZATION CODE ***
                                XFC01:; MOV     #DMREG2,BAR     %JMP    XFC01   ; [16K]EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                                ; [QDA]
003367  033716  000603  010000          MOV     (BUF),R16 @RDPF                 ; [16K]R16=SECTOR SIZE
003370  130456  000016  000000          INC     R16                             ; [16K]ADD ONE
003371  013740  007007  123611          MOV     #DMREG0,BAR     %CALL   S.LD17  ; R17=TABLE POINTER
003372  114557  000100  000000          BIT     #BIT14,R17                      ; IF RECIRCULATE BIT SET
003373  013460  010400  043374          NOP\T           @SDPF   %TNZRO          ; THEN SET DPF
003374  120357  007017  123607          INCB    R17\0,R17,BAR   %CALL   S.LDDB  ; DBAR=BUFFER POINTER
003375  120357  007017  133605          INCB    R17\0,R17,BAR   %CALL   S.LD13  ; R13=LO HEADER FIELD
003376  120357  007017  133606          INCB    R17\0,R17,BAR   %CALL   S.LD14  ; R14=HI HEADER FIELD
003377  013740  067005  133611          MOV     #DMREG5,BAR     %CDPF   S.LD17  ; IF RECIRCULATE THEN RELOAD R17
003400  013740  007007  123622          MOV     #DMREG0,BAR     %CALL   S.ST17  ; UPDATE TABLE POINTER
003401  013740  007001  123612          MOV     #DMREG1,BAR     %CALL   S.LL12  ; R12=HEAD ADDRESS
003402  033757  000040  000000          MOV     #IDX,R17                        ; [EC0#1]R17=INDEX PULSE FLAG
                                ; ***   WAIT FOR SECTOR PULSE (TO AVOID RACE IN DRIVE) ***
003403  033751  000060  132507          MOV     #<SEC+IDX>,R11  %CALL   D.CPUL  ; GO CHECK PULSE
003404  133552  000053  010000          OR      #FORICD,R12                     ; R12=FORMAT ON INDEX CODE & HEAD ADDR
                                ;.endc                                          ; [QDA]
                                ; ***   ENTRY POINT FOR MSCP REPLACE COMMAND ***
                                XF.REP:
```

```
                                ;.if    nz,qdadm                        ; [QDA]
003405  033751  000012  132513          MOV     #10.,R11        %CALL   D.WSIG  ; [16K]WAIT FOR RECVR RDY & R/W RDY
003406  013440  010520  143461          NOP\F           @SWM    %JZRO   XRWRER  ; [16K]IF ERROR THEN DONE/ELSE SET WRITE MODE
                                ; ***   START OF MAIN BODY OF FORMAT CODE ***
003407  013440  000440  000000  XFC1A:  NOP             @SCMD                   ; [16K]SET COMMAND MODE
003410  013740  005003  000000          MOV     #<CONT+WRT>,RTCS                ; [16K]SET WRITE GATE
003411  013740  007202  133055          MOV     #SYNC,BAR       %CALL   D.SRTC  ; [16K]BAR=SYNC/SEND REAL TIME COMMAND
                                ; ***   WAIT FOR INDEX/SECTOR PULSE TO GO HIGH ***
003412  014517  100502  053412          BIT\F   (RTDS),R17 @RSE %JNDCLK .       ; [16K]WHEN DRV CLK,CHK FOR PULSE HI
003413  014517  010002  103413          BIT     (RTDS),R17      %JZRO   .       ; [16K]WAIT FOR PULSE TO GO HI
                                                                                ; [ECO#2]D.SRTC RETURNS DMREG IN BAR
                                ; ***   WAIT FOR INDEX/SECTOR PULSE TO GO LOW ***
003414  014517  010002  003414          BIT     (RTDS),R17      %JNZRO  .       ; [16K]WAIT FOR PULSE TO GO LO
                                ; ***   GO WRITE HEADER AND DATA ***
003415  033712  000703  122550          MOV     (BUF),R12 @SSE  %CALL   D.WHDR  ; [16K]GO WRITE HEADER
003416  014440  106772  043416          CLR\F   R12,SD  @SECCT  %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#2(259)
003417  013740  007007  000000          MOV     #DMREG0,BAR                     ; BAR=DM REG 0 PTR
003420  033717  106003  043420          MOV\F   (BUF),R17,SD    %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#3(260)
003421  014440  106012  043421          CLR\F   R12,SD          %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#4(261)
003422  120357  007017  010000          INCB    R17\0,R17,BAR                   ; BAR=1ST WORD OF TABLE ENTRY
003423  033715  106003  053423          MOV\F   (BUF),DBAR,SD   %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#5(262)
003424  033451  106015  053424          MOV\F   DBAR,R11,SD     %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#6(263)
003425  120357  007017  010000          INCB    R17\0,R17,BAR                   ; BAR=2ND WORD OF ENTRY
003426  033713  106003  053426          MOV\F   (BUF),R13,SD    %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#7(264)
003427  120357  007017  010000          INCB    R17\0,R17,BAR                   ; BAR=3RD WORD OF ENTRY
003430  033714  106003  053430          MOV\F   (BUF),R14,SD    %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#8(265)
003431  013740  007005  010000          MOV     #DMREG5,BAR                     ; BAR=PTR TO RECIRCULATE ENTRY
003432  134551  106100  053432          AND\F   #BIT14,R11,SD   %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#9(266)
003433  033737  010003  053434          MOV\T   (BUF),R17       %TNZRO          ; IF RECIRC BIT SET THEN CHG R17
003434  133752  106115  043434          MOV\F   #FORSCD,R12,SD  %JNDCLK .       ; WHEN DRV CLK,(SD)=ECC#10(267)
003435  013740  007002  000000          MOV     #DMREG2,BAR                     ; [16K]BAR=PTR TO SECTOR SIZE
003436  034451  106011  053436          CLR\F   R11,R11,SD      %JNDCLK .       ; [16K]WHEN DRV CLK,(SD)=ECC#11(268)
003437  033716  000003  010000          MOV     (BUF),R16                       ; [16K]R16=SECTOR SIZE
                                ; ***   IMPORTANT - RESET ECC TIME (@RECCT) ON DATA WORD 269, WORDS 269 & 270 MUST BE ZERO ***
003440  034451  106571  043440          CLR\F   R11,R11,SD @RECCT %JNDCLK .     ; WHEN DRV CLK,(SD)=ECC#12(269)
003441  130456  000016  000000          INC     R16                             ; [16K]ADD ONE TO SECTOR SIZE
003442  034451  106011  053442          CLR\F   R11,R11,SD      %JNDCLK .       ; WHEN DRV CLK,(SD)=0(270)
003443  013740  007007  000000          MOV     #DMREG0,BAR                     ; [16K]BAR=DM REG 0 PTR
003444  034451  106011  053444          CLR\F   R11,R11,SD      %JNDCLK .       ; WHEN DRV CLK,(SD)=0(271)
003445  013440  163017  103462          MOV     R17,BUF         %JLATE  XFC1ER  ; [ECO#2]UPDATE TABLE ENTRY PTR IN DM REG 0
003446                                  ASSUME  CONT,EQ,1                       ; MAKE SURE CONT IS 1
                                ; ***   RESET WRITE GATE ON WORD 272. - (R11 INCREMENTED = #CONT) ***
                                ; ***   RESET ECC ON WORD 272. ***
003446  110440  105551  043446          INC\F   R11,RTCS @RECC  %JNDCLK .       ; WHEN DRV CLK,(SD)=WORD 272
003447  013440  140000  113462          NOP                     %JDSER  XFC1ER  ; [ECO#2]IF RTDS PULSE/PARITY ERROR THEN DONE
003450  037151  000511  122513          XNOR    R11,R11 @RSE %CALL      D.WSIG  ; [ECO#2]RESET SERDES/WAIT FOR RCVR RDY & R/W RDY
003451  013455  010015  103461          TST     DBAR            %JZRO   XRWRER  ; [ECO#2]IF NOT SET THEN ERROR
003452  033757  030020  003407          MOV     #SEC,R17        %JNMSB  XFC1A   ; IF BIT15 NOT SET THEN CONTINUE
                                ; ***   END OF DM FORMAT ROUTINE ***
003453  034451  000011  000000          CLR     R11                             ; [16K]R11=0 (SUCCESS)
                                ;.endc                                  ; [QDA]
                                ; ***   ALL XFC'S EXIT THROUGH THIS CODE ***
                                XFCSST:
                                ;.if    nz,qdadm                        ; [QDA]
```



```
003454  013440  000540  121270          NOP             @RECC   %CALL   D.CLCS  ; RESET ECC TIMING IN CASE NOT DONE
003455  013440  000500  000000          NOP             @RSE                    ; RESET SERDES
                                ; ***   CHECK FOR REPLACE COMMAND ***
003456  114544  000100  010000          BIT     #DMODE,RLL                      ; IF NOT IN DM MODE
003457  013740  017001  137777          MOV     #DMREG1,BAR     %RZRO           ;   THEN MUST BE REPLACE COMMAND/RETURN
003460  013440  003011  104006          MOV     R11,BUF         %JMP    XFCRET  ; STORE R11/GO BACK TO DM
                                ; ***   FORMAT ERROR EXITS ***
003461  033751  000002  113454  XRWRER: MOV     #2,R11          %JMP    XFCSST  ; READ/WRITE READY ERROR
003462  033751  000001  113454  XFC1ER: MOV     #1,R11          %JMP    XFCSST  ; R11=FAILURE CODE
                                ;.endc                                  ; [QDA]
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       XFC02
                                ;       XFC03
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL PROCESS THE READ/WRITE N SECTORS XFC'S. THIS
                                ; COMMAND WILL ENABLE A DM PROGRAM TO READ/WRITE ANY SECTOR(S) ON THE DISK.
                                ; THE DM PROGRAM MUST COMPLETE ALL HEAD POSITIONING REQUIRED BEFORE EXECUTING
                                ; THIS COMMAND. DM REGISTER 0 POINTS TO THE FIRST BUFFER OF A BUFFER CHAIN
                                ; (THE BUFFERS ARE MIRRORS OF THE UDA SECTOR DATA BUFFERS), AND DM REGISTER 2
                                ; CONTAINS THE SECTOR SIZE IN WORDS.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO FIRST BUFFER IN CHAIN
                                ;                               MSB = 1 - NO REVECTORING PLEASE
                                ;       DM REGISTER 2           SECTOR SIZE IN WORDS
                                ;
                                ; OUTPUTS:
                                ;       EMPTIED/FILLED DATA BUFFERS ON SUCCESSFUL COMPLETION
                                ;       DM REGISTER 1           0 - SUCCESS
                                ;                               1 - DRIVE SIGNAL TIMEOUT
                                ;
                                ;;-

                                XFC02: ;CLR     R16             %JMP    XFCRW   ; EXECUTED IN JUMP TABLE
                                XFC03: ;MOV     #1,R16          %JMP    XFCRW   ; EXECUTED IN JUMP TABLE
                                XFCRW:
                                ;.if    nz,qdadm                                ; [QDA]
003463  013440  000600  000000          NOP             @RDPF                   ; [16K]RESET DPF
003464  013740  007007  123611          MOV     #DMREGO,BAR     %CALL   S.LD17  ; R17=BUFFER POINTER
003465  013460  030400  153466          NOP\T           @SDPF   %TMSB           ; IF MMSB SET THEN NO REVECTORS
003466  013456  000016  000000          TST     R16                             ; IF READ XFC
003467  013440  010000  112606          NOP                     %JZRO   D.REDX  ;   THEN GO DO XFC READ
003470  013440  000000  112130          NOP                     %JMP    D.WRTX  ;   ELSE GO DO XFC WRITE
                                ;.endc                                          ; [QDA]
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC04
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE SEND SDI COMMAND XFC. DM REGISTER 0
                                ; CONTAINS A POINTER TO A BUFFER INCLUDING THE COMMAND AND DATA, DM REGISTER
                                ; 1 CONTAINS THE BYTE COUNT FOR THE BUFFER (INCLUDING OP CODE). IF THE BYTE
                                ; COUNT IS ZERO THE FIRST WORD OF THE BUFFER WILL BE SENT AS A SDI REAL TIME
                                ; COMMAND.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO SDI COMMAND AND DATA BUFFER
                                ;       DM REGISTER 1           NUMBER OF BYTES TO TRANSFER FROM BUFFER
                                ;       DM REGISTER 2           SDI INTERCONNECT SELECT
                                ;
                                ; OUTPUTS:
                                ;       DATA TRANSFERRED TO DRIVE
                                ;       DM REGISTER 1           0 - SUCCESSFUL COMPLETION
                                ;                               1 - DRIVE SIGNAL TIMEOUT
                                ;
                                ;;-

                                XFC04: ;MOV     #DMREGO,BAR     %JMP    XFC04   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                                ; [QDA]
003471  033716  000003  131004          MOV     (BUF),R16       %CALL   X.SSDI  ; GO SELECT SDI INTERCONNECT
                                                                                ; R16=POINTER TO COMMAND BUFFER
003472  013440  000000  133474          NOP                     %CALL   XF04A   ; GO DO THE RIGHT COMMAND
003473  013440  000000  113454          NOP                     %JMP    XFCSST  ; SET DM R1/GO BACK TO DM
003474  013740  007001  123611  XF04A:  MOV     #DMREG1,BAR     %CALL   S.LD17  ; R17=# BYTES TO TRANSFER
003475  013440  017016  051347          MOV\F   R16,BAR         %JNZRO  LEV2.T  ; IF BYTE CNT NEQ 0 THEN SEND COMMAND FROM TABLE
003476  033712  000003  111212          MOV     (BUF),R12       %JMP    LEVOWR  ;   ELSE GO SEND REAL TIME COMMAND
                                ;.endc                                          ; [QDA]
                                        .PAGE
```

```
                                        ;;+
                                        ; ROUTINE NAME:
                                        ;       XFC05
                                        ;
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;       THIS ROUTINE WILL PROCESS THE RECEIVE SDI RESPONSE XFC. THIS ALLOWS
                                        ; THE PROGRAM RUNNING ON THE DM TO GET A RESPONSE TO A SDI COMMAND SENT VIA
                                        ; THE SEND SDI COMMAND XFC. DM REGISTER 0 CONTAINS A POINTER TO A BUFFER
                                        ; WHICH IS TO HOLD THE DRIVE RESPONSE, DM REGISTER 1 CONTAINS THE SIZE OF
                                        ; THE BUFFER DESCRIBED BY R0, AND DM REGISTER 2 CONTAINS THE SDI INTERCONNECT
                                        ; SELECT.
                                        ;
                                        ; INPUTS:
                                        ;       DM REGISTER 0           POINTER TO DRIVE RESPONSE BUFFER
                                        ;       DM REGISTER 1           NUMBER OF WORDS IN BUFFER
                                        ;       DM REGISTER 2           SDI INTERCONNECT SELECT
                                        ;
                                        ; OUTPUTS:
                                        ;       FILLED DRIVE RESPONSE BUFFER
                                        ;       DM REGISTER 0           DRIVE RESPONSE OP CODE
                                        ;       DM REGISTER 1           BIT00 - TIMEOUT OF RECEIVE OPERATION
                                        ;                               BIT01 - FIRST WORD NOT A START FRAME
                                        ;                               BIT02 - FRAMING ERROR
                                        ;                               BIT03 - CHECKSUM ERROR
                                        ;
                                        ;;-

                                        XFC05: ;NOP                             %JMP    XFC05   ; EXECUTED IN JUMP TABLE
                                        ;.if    nz,qdadm                                        ; [QDA]
003477  133755  000024  131004                MOV     #12000,R15      %CALL   X.SSDI  ; GO SELECT SDI INTERCONNECT
003500  013740  007007  123605  XFC5A:  MOV     #DMREG0,BAR     %CALL   S.LD13  ; R13=POINTER TO DRIVE RESPONSE BUFFER
003501  013740  007001  123606          MOV     #DMREG1,BAR     %CALL   S.LD14  ; R14=SIZE OF BUFFER IN WORDS
003502  013440  000000  131273          NOP                     %CALL   LEV1RD  ; GO TRY AND READ DRIVE RESPONSE
003503  014551  000001  010000          BIT     #DCERR,R11                      ; IF NOT DRIVE CLK TIMEOUT
003504  031455  010015  153506          DEC\F   R15             %JZRO   XFC5B   ;   THEN RESPONSE RECEIVED
003505  013440  010000  013500          NOP                     %JNZRO  XFC5A   ; IF R15 NEQ 0 THEN LOOP
003506  013740  007007  133615  XFC5B:  MOV     #DMREG0,BAR     %CALL   S.ST12  ; STORE OP CODE IN DM REG 0
003507  013440  000000  113454          NOP                     %JMP    XFCSST  ; SET DM R1/GO BACK TO DM
                                        ;.endc                                  ; [QDA]
                                                .PAGE
```



```
                                        ;;+
                                        ; ROUTINE NAME:
                                        ;       XFC06
                                        ;
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;       THIS ROUTINE WILL PERFORM THE COMPARE DATA PATTERN TO BUFFER XFC.
                                        ; DM REGISTER 0 CONTAINS THE WORD COUNT, DM REGISTER 1 CONTAINS THE
                                        ; BUFFER START ADDRESS, DM REGISTER 2 CONTAINS THE FIRST DATA PATTERN,
                                        ; DM REGISTER 3 CONTAINS THE SECOND DATA PATTERN, AND DM REGISTER 4 CONTAINS
                                        ; THE THIRD DATA PATTERN. THE FIRST DATA PATTERN WILL
                                        ; COMPARED TO THE FIRST BUFFER WORD, THE SECOND TO THE SECOND, THE THIRD
                                        ; TO THE THIRD, THEN THE FIRST TO THE FOURTH, THE SECOND TO THE FIFTH,
                                        ; ETC., ETC.
                                        ;
                                        ; INPUTS:
                                        ;       DM REGISTER 0           POINTER TO TABLE CONTAINING THE FOLLOWING:
                                        ;       WORD 0 OF TABLE         WORD COUNT
                                        ;       WORD 1 OF TABLE         POINTER TO START OF COMPARE BUFFER
                                        ;       WORD 2 OF TABLE         COMPARE PATTERN #1
                                        ;       WORD 3 OF TABLE         COMPARE PATTERN #2
                                        ;       WORD 4 OF TABLE         COMPARE PATTERN #3
                                        ;
                                        ; OUTPUTS:
                                        ;       DM REGISTER 1           0 - ALL TRIPLE WORDS OF THE BUFFER COMPARED
                                        ;                                   TO THE TRIPLE WORD COMPARE PATTERNS
                                        ;                               NONZERO - COMPARE ERROR AND VALUE IS OFFSET
                                        ;                               OF TRIPLE WORD FROM THE END OF THE BUFFER.
                                        ;
                                        ;;-

                                        XFC06: ;NOP                             %JMP    XFC06   ; EXECUTED IN JUMP TABLE
                                        ;.if    nz,qdadm                                        ; [QDA]
003510  013740  007007  133604                MOV     #DMREG0,BAR     %CALL   S.LD12  ; R12=TABLE POINTER
003511  010552  007001  133606                ADD     #1,R12\N,BAR    %CALL   S.LD14  ; R14=BUFFER START ADDRESS
003512  010552  007002  123607                ADD     #2,R12\N,BAR    %CALL   S.LD15  ; R15=DATA PATTERN #1
003513  010552  007003  133610                ADD     #3,R12\N,BAR    %CALL   S.LD16  ; R16=DATA PATTERN #2
003514  010552  007004  133611                ADD     #4,R12\N,BAR    %CALL   S.LD17  ; R17=DATA PATTERN #3
003515  010552  007000  123603                ADD     #0,R12\N,BAR    %CALL   S.LD11  ; R11=WORD COUNT
003516  120354  017014  113454  XFC06A: INCB    R14\0,R14,BAR   %JZRO   XFCSST  ; BAR=ADDR/IF LAST ALU EQ 0 THEN RETURN
003517  016515  000003  010000          XOR     (BUF),R15\N                     ; IF BUFFER NEQ PATTERN #1
003520  120354  017014  003454          INCB    R14\0,R14,BAR   %JNZRO  XFCSST  ;   THEN ERROR/ELSE BAR=NEXT WORD/INCR R14
003521  031451  000011  000000          DEC     R11                             ; DECR WORD COUNT
003522  016516  010003  143454          XOR\F   (BUF),R16\N     %JZRO   XFCSST  ; IF WORD CNT EQ 0 THEN DONE/IF BUFFER EQ PATTERN #2
003523  120354  017014  003454          INCB    R14\0,R14,BAR   %JNZRO  XFCSST  ;   THEN ERROR/ELSE BAR=NEXT WORD/INCR R14
003524  031451  000011  000000          DEC     R11                             ; DECR R11
003525  016517  010003  153454          XOR\F   (BUF),R17\N     %JZRO   XFCSST  ; IF BUFFER EQ PATTERN #3
003526  031471  010011  153516          DEC\T   R11             %JZRO   XFC06A  ;   THEN DECR R11 & LOOP
003527  013440  000000  113454          NOP                     %JMP    XFCSST  ;   ELSE ERROR AND EXIT
                                        ;.endc                                  ; [QDA]
                                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC07
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE RETURN DRIVE SIGNALS XFC. DM REGISTER
                                ; 2 CONTAINS THE SDI INTERCONNECT TO SELECT. THE DRIVE SIGNAL STATUS WILL
                                ; BE RETURNED IN DM REGISTER 1.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 2           SDI INTERCONNECT SELECT
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 1           REAL TIME DRIVE STATUS
                                ;
                                ;;-

                                XFC07: ;NOP                    %JMP    XFC07   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                        ; [QDA]
003530  013440  000000  121004          NOP                     %CALL   X.SSDI  ; GO SELECT SDI INTERCONNECT
003531  033711  000002  010000          MOV     (RTDS),R11              ; [QDA]SET DM R1
                                            ; CLEAR THESE PREVIOUSLY FLOATING BITS
003532  035551  000214  010000          BIC     #<BIT02+BIT03+BIT07>,R11        ; [QDA] DM BUG FIX
003533  135551  000001  000000          BIC     #<BIT08>,R11            ; [QDA] DM BUG FIX
                                ;
                                ;       MUST SET BITS IN RTDS THAT HAVE BEEN REMOVED AND ARE ONLY JUMP CONDITIONS
                                ;
003534  033571  110004  143535          BIS\T   #BIT02,R11      %TCSR           ; [QDA]SET RTCS RDY H
003535  033571  150010  153536          BIS\T   #BIT03,R11      %TDSR           ; [QDA]SET RTDS RCVD H
003536  033571  140200  153537          BIS\T   #BIT07,R11      %TDSER          ; [QDA]SET DATA XMIT ERROR H
003537  133571  140001  143540          BIS\T   #BIT08,R11      %TDSER          ; [QDA]SET RECV ERROR H
003540  013440  000000  113454          NOP                     %JMP    XFCSST  ; [QDA]RETURN BACK TO DM
                                ;.endc                                  ; [qda]
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC08
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE ECHO DATA TO DRIVE XFC. DM
                                ; REGISTER 0 CONTAINS THE DATA TO BE ECHOED TO THE DRIVE AND DM REGISTER
                                ; 2 CONTAINS THE SDI INTERCONNECT SELECT CODE.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           DATA TO ECHO TO DRIVE (HI BYTE MUST BE ZERO!)
                                ;       DM REGISTER 2           SDI INTERCONNECT SELECT
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 0           WORD OF DATA ECHOED BACK FROM DRIVE
                                ;       DM REGISTER 1           0 - SUCCESSFUL
                                ;                               1 - TIMEOUT ON SDI LEVEL 1 SEND
                                ;                               2 - TIMEOUT ON SDI LEVEL 1 RECEIVE
                                ;
                                ;;-

                                XFC08: ;MOV    #DMREG0,BAR     %JMP    XFC08   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                        ; [QDA]
003541  033712  000003  121004          MOV     (BUF),R12       %CALL   X.SSDI  ; SELECT SDI INTERCONNECT/R12=DATA TO ECHO
003542  133552  000350  131212          BIS     #ECHOCD,R12     %CALL   LEVOWR  ; SEND ECHO REAL TIME COMMAND
003543  033753  010377  043454          MOV\F   #377,R13        %JNZRO  XFCSST  ; IF TIMEOUT THEN EXIT/ELSE R13=RECV T/O VALUE
003544  013740  007007  121244  XFC08A: MOV     #DMREG0,BAR     %CALL   LEVORD  ; DO LEVEL 0 READ/RESET BAR
003545  031453  010013  103547          DEC     R13             %JZRO   XFC08B  ; IF 0 THEN DONE/ELSE DECR R13
003546  130451  010011  043544          INC\F   R11             %JNZRO  XFC08A  ; DO TIMEOUT LOOP
003547  013440  003012  113454  XFC08B: MOV     R12,BUF         %JMP    XFCSST  ; SET DM R1/GO BACK TO DM
                                ;.endc                                  ; [QDA]
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC09
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE INITIALIZE DRIVE XFC COMMAND. THE
                                ; SDI INTERCONNECT SELECT WILL BE CONTAINED IN DM REGISTER 2.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 2           SDI INTERCONECT SELECT
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;;-

                                XFC09: ;NOP                     %JMP    XFC09   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                                ; [QDA]
003550  013440  000000  121004          NOP                     %CALL   X.SSDI  ; GO SELECT SDI INTERCONNECT
003551  013440  000000  131022          NOP                     %CALL   D.INIT  ; GO INIT DRIVE
003552  013440  000000  104006          NOP                     %JMP    XFCRET  ; GO BACK TO DM
                                ;.endc                                  ; [QDA]
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC10
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PROCESSES THE WAIT FOR SECTOR/INDEX PULSE XFC.
                                ; THIS ROUTINE WILL WAIT FOR THE LEADING EDGE OF SECTOR OR INDEX
                                ; PULSE TO OCCUR AND THEN WAIT FOR THE TRAILING EDGE BEFORE RETURNING
                                ; TO THE CALLING PROGRAM.
                                ;
                                ; INPUTS:
                                ;       DMREG2          SDI INTERCONNECT
                                ;
                                ;;-

                                XFC10: ;NOP                     %JMP    XFC10   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                                ; [QDA]
003553  013440  000000  121004          NOP                     %CALL   X.SSDI  ; GO SELECT SDI INTERCONNECT
003554  033751  000060  132507          MOV     #<SEC+IDX>,R11  %CALL   D.CPUL  ; GO WAIT FOR SECTOR/INDEX TRANSITION
003555  034451  000011  113454          CLR     R11             %JMP    XFCSST  ; ZAP R11/EXIT XFC
                                ;.endc                                  ; [QDA]
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC11
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE READ UNIBUS MEMORY XFC. THIS ALLOWS
                                ; THE DM PROGRAM TO READ DATA FROM THE HOST. DM REGISTER 0 CONTAINS THE LOW
                                ; ORDER 16 BITS OF THE UNIBUS ADDRESS, DM REGISTER 1 CONTAINS THE HIGH ORDER
                                ; 16 BITS OF THE UNIBUS ADDRESS, DM REGISTER 2 CONTAINS THE NUMBER OF WORDS
                                ; TO TRANSFER, AND DM REGISTER 3 CONTAINS THE DM BUFFER ADDRESS. WORD COUNT
                                ; AND BUFFER ADDRESS VALIDATION ARE PERFORMED HERE (D.PROC) AND A FLAG IS SET
                                ; TO REQUEST XFC SERVICE FROM THE U.PROCESSOR. THE XFC CODE IS PASSED TO THE
                                ; U.PROC IS REGISTER R11 AND THE D.PROC THEN SETS THE REQUEST XFC SERVICE BIT
                                ; IN THE RLL (SYSTEM STATUS) REGISTER. THE U.PROCESSOR CLEARS THIS BIT WHEN
                                ; THE XFC IS COMPLETE.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           LO ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 1           HI ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 2           NUMBER OF WORDS TO TRANSFER
                                ;       DM REGISTER 3           DM DATA BUFFER ADDRESS
                                ;
                                ; OUTPUTS:
                                ;       DATA READ FROM THE HOST
                                ;       DM REGISTER 1                   0 - SUCCESSFUL
                                ;                                       1 - WORD COUNT EXCEEDS 16K
                                ;                                       2 - DM BUFFER ADDRESS LESS THAN UN.BUF
                                ;                                       4 - NONEXISTENT UNIBUS MEMORY
                                ;                                       8 - UNIBUS PARITY ERROR
                                ;
                                ;;-

                                XFC11: ;MOV     #11.,R11        %JMP    XFC11   ; EXECUTED IN JUMP TABLE
                                ;.if    nz,qdadm                                ; [QDA]
003556  013740  007002  133604          MOV     #DMREG2,BAR     %CALL   S.LD12  ; R12:# WORDS TO TRANSFER
003557  114552  000300  010000          BIT     #140000,R12                     ; [16K]TEST FOR GREATER THAN 16K
003560  033771  010001  043564          MOV\T   #BIT00,R11      %JNZRO  XF11C   ; IF TOO BIG THEN ERROR EXIT
003561  133544  000020  010000  XF11A:  BIS     #DXFC,RLL                       ;   ELSE OK AND TELL U.PROC
003562  114544  000020  010000  XF11B:  BIT     #DXFC,RLL                       ; CHECK IF U.PROC DONE
003563  114544  010100  013562          BIT     #DMODE,RLL      %JNZRO  XF11B   ; IF NOT DONE THEN LOOP
003564  013440  010000  013454  XF11C:  NOP                     %JNZRO  XFCSST  ; IF IN DM MODE THEN EXIT
003565  135544  000040  100701  XF11D:  BIC     #DMBEG,RLL      %JMP    D.IDLE  ;   ELSE MUST HAVE BEEN XFC 17
                                ;.endc                                  ; [QDA]
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       XFC12
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS THE WRITE UNIBUS MEMORY XFC. THIS ALLOWS
                                ; THE DM PROGRAM TO WRITE DATA FROM THE HOST. DM REGISTER 0 CONTAINS THE LOW
                                ; ORDER 16 BITS OF THE UNIBUS ADDRESS, DM REGISTER 1 CONTAINS THE HIGH ORDER
                                ; 16 BITS OF THE UNIBUS ADDRESS, DM REGISTER 2 CONTAINS THE NUMBER OF WORDS
                                ; TO TRANSFER, AND DM REGISTER 3 CONTAINS THE DM BUFFER ADDRESS. WORD COUNT
                                ; AND BUFFER ADDRESS VALIDATION ARE PERFORMED HERE (D.PROC) AND A FLAG IS SET
                                ; TO REQUEST XFC SERVICE FROM THE U.PROCESSOR. THE XFC CODE IS PASSED TO THE
                                ; U.PROC IS REGISTER R11 AND THE D.PROC THEN SETS THE REQUEST XFC SERVICE BIT
                                ; IN THE RLL (SYSTEM STATUS) REGISTER. THE U.PROCESSOR CLEARS THIS BIT WHEN
                                ; THE XFC IS COMPLETE.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           LO ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 1           HI ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 2           NUMBER OF WORDS TO TRANSFER
                                ;       DM REGISTER 3           DM DATA BUFFER ADDRESS
                                ;
                                ; OUTPUTS:
                                ;       DATA WRITTEN TO THE HOST
                                ;       DM REGISTER 1                   0 - SUCCESSFUL
                                ;                                       1 - WORD COUNT EXCEEDS 4K
                                ;                                       2 - DM BUFFER ADDRESS LESS THAN 640
                                ;                                       4 - NONEXISTENT UNIBUS MEMORY
                                ;
                                ;;-

                                XFC12: ; MOV    #12.,R11        %JMP    XFC11   ; EXECUTED IN JUMP TABLE
                                        .PAGE
```

```
                                .SBTTL  U.PROC AND D.PROC SUBROUTINES
                              ;;+     04-FEB-83       USA MICROCODE
                              ; ROUTINE NAMES:
                              ;       P.LOCK
                              ;       UNLOCK
                              ;
                              ; FUNCTIONAL DESCRIPTION:
                              ;       P.LOCK WILL WAIT FOR THE ALTERNATE PROCESSOR TO SET THE 'PLOCK' BIT
                              ; IN THE CONTROL REGISTER IMAGE (RLL). IT WILL THEN CLEAR THE 'PLOCK' BIT
                              ; IN THE RLL TO LOCK OUT THE ALTERNATE PROCESSOR.
                              ;       UNLOCK WILL SET THE 'PLOCK' BIT IN RLL, EFFECTIVELY UNLOCKING THE
                              ; ALTERNATE PROCESSOR.
                              ;
                              ; INPUTS:
                              ;       RLL             CONTROL REGISTER IMAGE
                              ;
                              ; OUTPUTS:
                              ;       P.LOCK - ALTERNATE PROCESSOR LOCKED OUT OF MUTEXED DATA AREAS
                              ;       UNLOCK - ALTERNATE PROCESSOR FREE TO USE MUTEXED DATA AREAS
                              ;
                              ; STACK LEVEL:
                              ;       NONE USED
                              ;
                              ;;-

003566                                ASSUME  PLOCK,EQ,BIT15                  ; PLOCK MUST BE MSB
003566  013440  020000  137777  P.LCKA: NOP                     %RCRY           ; IF OTHER PROC FREE THEN RETURN
003567  050144  000004  103566  P.LOCK: ADD\R   RLL,RLL          %JMP    P.LCKA  ; IF OTHER PROCESSOR MUTEXING

003570  133544  000200  137777  UNLOCK: BIS     #PLOCK,RLL      %RET            ; SET PLOCK/RETURN
                                .PAGE
```



```
                              ;;+
                              ; ROUTINE NAME:
                              ;       D.GMST (D.PROC GET MUTEXED SDI STATUS)
                              ;
                              ; FUNCTIONAL DESCRIPTION:
                              ;       THESE ROUTINES WILL LOCK OUT THE OPPOSITE PROCESSOR (USING THE
                              ; PLOCK BIT IN RLL), LOAD THE SDI STATUS (FROM SDI.ST) INTO R0 OR R11,
                              ; AND RETURN. IT SHOULD BE NOTED THAT IT IS THE CALLER'S RESPONSIBILITY
                              ; TO FREE THE OPPOSITE PROCESSOR
                              ; BY EITHER CALLING UNLOCK OR BY PERFROMING THE FOLLOWING INSTRUCTION:
                              ;       BIS     #PLOCK,RLL
                              ;
                              ; INPUTS:
                              ;       DBAR POINTING TO SDI CONTROL BLOCK OF INTEREST
                              ;
                              ; OUTPUTS:
                              ;       SDI STATUS IN R11
                              ;       LOCKED OUT U.PROC OR D.PROC
                              ;
                              ; STACK LEVEL:
                              ;       NONE USED
                              ;
                              ;;-

003571  010555  027000  113603          ADD     #SDI.ST,DBAR\N,BAR %JCRY S.LD11 ; THEN R11 = SDI STATUS / RETURN
003572  050144  000004  103571  D.GMST: ADD\R   RLL,RLL          %JMP    .-1     ; IF PLOCK (BIT15) IS ONE
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.ROTL  (ROTATE R12 LEFT)
                                ;       S.SWAB  (SWAP BYTES OF R12)
                                ;       S.RR17  (ROTATE R17 RIGHT)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       S.ROTL WILL ROTATE R12 LEFT THE NUMBER OF TIMES SPECIFIED
                                ; IN R11. THIS IS THE DRIVE PROCESSOR'S ROTATE LEFT ROUTINE.
                                ;
                                ;       S.SWAB WILL SWAP THE BYTES OF R11
                                ;
                                ;       S.RR17 WILL ROTATE R17 RIGHT
                                ;
                                ; INPUTS:
                                ;       R11 (S.ROTL, S.RR17)            THE NUMBER OF TIMES TO ROTATE LEFT
                                ;       R12 (S.ROTL)                    THE REGISTER TO ROTATE LEFT
                                ;       R17 (S.RR17)                    THE REGISTER TO ROTATE RIGHT
                                ;
                                ; OUTPUTS:
                                ;       R12             ROTATED LEFT R11 TIMES / BYTE-SWAPPED
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

003573  073452  010012  137777          SHF\L   R12             %RZRO           ; ROTATE R12 LEFT/IF R11 = 0 THEN RETURN
003574  031451  000011  103573  S.ROTL: DEC     R11             %JMP    .-1     ; DECR R11
003575  033751  000010  113574  S.SWAB: MOV     #8.,R11         %JMP    S.ROTL  ; SWAP BYTES ENTRY

                                ;       ROUTINE TO SHIFT R17 RIGHT THE NUMBER OF PLACES SPECIFIED BY R11

003576  053457  010017  127777          SHF\R   R17             %RZRO
003577  031451  000011  103576  S.RR17: DEC     R11             %JMP    .-1
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.SD11  (GETS ADDRESS OF SDI.1 IN R11)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINES WILL LOAD THE ADDRESS OF SDI.1 IN R11.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       R11             ADDRESS OF SDI.1
                                ;
                                ;;-

003600  033751  000355  010000  S.SD11: MOV     #SDI.1&LOBYT,R11                ; R11=LO SDI.1 ADDRESS
003601  133551  000002  127777          BIS     #SDI.1&HIBYT,R11 %RET           ; SET IN HI ADDRESS/RETURN
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       S.LDCR,S.LD00,ETC.
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL LOAD THE REGISTER NUMBER SPECIFIED AS THE LAST
                                ; TWO CHARACTERS OF THE SUBROUTINE NAME, I.E. S.LD11 WILL LOAD R11 FROM BUF.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO THE BUFFER ADDRESS WHOSE DATA IS DESIRED
                                ;                       TO BE LOADED IN THE REGISTER
                                ;
                                ; OUTPUTS:
                                ;       REGISTER LOADED WITH CONTENTS OF BUFFER POINTED TO BY BAR
                                ;
                                ;;-

003602  033702  000003  137777  S.LDR2: MOV     (BUF),R2        %RET            ; LOAD R2 FROM BUFFER
003603  033711  000003  127777  S.LD11: MOV     (BUF),R11       %RET            ; LOAD R11 FROM BUFFER
                                S.LDPC:
003604  033712  000003  127777  S.LD12: MOV     (BUF),R12       %RET            ; LOAD R12 FROM BUFFER
003605  033713  000003  137777  S.LD13: MOV     (BUF),R13       %RET            ; LOAD R13 FROM BUFFER
                                S.LDCC:
003606  033714  000003  127777  S.LD14: MOV     (BUF),R14       %RET            ; LOAD R14 FROM BUFFER
                                S.LDCY:
                                S.LD15:
003607  033715  000003  137777  S.LDDB: MOV     (BUF),R15       %RET            ; LOAD R15/DBAR FROM BUFFER
                                LDRT1:
003610  033716  000003  137777  S.LD16: MOV     (BUF),R16       %RET            ; LOAD R16 FROM BUFFER
003611  033717  000003  127777  S.LD17: MOV     (BUF),R17       %RET            ; LOAD R17 FROM BUFFER

                                ; ***   SPECIALTY ROUTINES ***

003612  033712  000003  000000  S.LL12: MOV     (BUF),R12                       ; R12:=(BUF)
003613  034552  000377  137777  S.ML12: AND     #LOBYT,R12      %RET            ; ISOLATE LO BYTE/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       S.STQQ,S.STRO,ETC.
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL STORE THE THE REGISTER NUMBER SPECIFIED AS THE
                                ; LAST TWO CHARACTERS OF THE SUBROUTINE NAME INTO THE BUFFER LOCATION POINTED
                                ; TO BY BAR, I.E. S.ST11 WILL STORE R11 INTO THE BUFFER.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO BUFFER AREA WHERE REGISTER DATA IS
                                ;                       DESIRED TO BE STORED
                                ;
                                ;;-

003614  013440  003011  127777  S.ST11: MOV     R11,BUF         %RET            ; STORE R11 INTO BUFFER
                                S.STPC:
003615  013440  003012  127777  S.ST12: MOV     R12,BUF         %RET            ; STORE R12 INTO BUFFER
003616  013440  003013  137777  S.ST13: MOV     R13,BUF         %RET            ; STORE R13 INTO BUFFER
                                S.STCC:
003617  013440  003014  127777  S.ST14: MOV     R14,BUF         %RET            ; STORE R14 INTO BUFFER
                                S.ST15:
003620  013440  003015  137777  S.STDB: MOV     R15,BUF         %RET            ; STORE R15/DBAR INTO BUFFER
                                STORT1:
003621  013440  003016  137777  S.ST16: MOV     R16,BUF         %RET            ; STORE R16 INTO BUFFER
003622  013440  003017  127777  S.ST17: MOV     R17,BUF         %RET            ; STORE R17 INTO BUFFER
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.CLRB  (CLEAR BUFFER LOCATION)
                                ;       INIT1   (SET BUFFER LOCATION TO 1)
                                ;       INITM1  (SET BUFFER LOCATION TO -1)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THE S.CLRB ROUTINE WILL STORE ZERO INTO THE BUFFER POINTED TO
                                ; BY BAR.
                                ;       THE INIT1 ROUTINE PUTS A 1 IN THE MEMORY LOCATION POINTED TO BY BAR.
                                ;       THE INITM1 ROUTINE PUTS A -1 IN THE MEMORY LOCATION POINTED TO BY BAR.
                                ;       ***NOTE THESE ROUTINES CAN BE CALLED BY EITHER PROCESSOR BECAUSE THEY
                                ; USE NO REGISTERS.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO BUFFER LOCATION TO CLEAR/SET TO 1/-1
                                ;
                                ; OUTPUTS:
                                ;       CLEARED BUFFER LOCATION (S.CLRB)
                                ;       1 IN BUFFER LOCATION (INIT1)
                                ;       -1 IN BUFFER LOCATION (INITM1)
                                ;
                                ;;-

003623  014440  003000  137777  S.CLRB: CLR     ,\N,BUF         %RET            ; (BUF) = 0/RETURN

003624  013740  003001  137777  INIT1:  MOV     #1,BUF          %RET            ; INITIALIZE TO 1/RETURN

003625  017740  003000  137777  INITM1: COM     #0,BUF          %RET            ; INITIALIZE TO 177777 / RETURN
                                .PAGE
```



```
                                .SBTTL  DIAGNOSTIC MACHINE INTERPRETER AND DMDT

                                ;
                                ;       R. LARY         27-JUN-79, UPDATED 03-DEC-80
                                ;       B. GRACE        SEPT-81, FOR [ECO#1]
                                ;       B. GRACE        NOV-81, FOR [16K]
                                ;       B. GRACE        FEB-82, FOR UDA52
                                ;
                                ;       DIAGNOSTIC MACHINE INSTRUCTION FORMATS
                                ;
                                ;       ARITHMETIC INSTRUCTIONS         OP + MODE + RB*10 + RA
                                ;                                       (RA=0 --> ABS MEM DEST IN LINE)

                                ;       OP                      MODE
                                ;
                                ;       100000  STORE           0       REGISTER
                                ;       101000  BITSET          200     REGISTER (RB) AUTOINC (270 = IMMEDIATE)
                                ;       102000  BITTEST         400     AUTODEC REGISTER (RB) (470 IS AN ILLEGAL MODE)
                                ;       103000  BITCLR          600     MEMORY, INDEXED BY RB (670 IS AN ILLEGAL MODE)
                                ;       104000  LOAD
                                ;       105000  ADD             100     INDICATES INDIRECTION
                                ;       106000  COMPARE
                                ;       107000  SUBTRACT                ** STORE ILLEGAL IN MODE 27X OR XX0 **
                                ;       114000  CLEAR (RA)             \
                                ;       115400  INCREMENT (RA)          \       "MODE" AND "RB" DON'T APPLY
                                ;       115000  TEST (RA)              /
                                ;       117400  DECREMENT (RA)         /
                                ;       110200  ROTATE (RA) LEFT       \
                                ;       110600  ROTATE (RA) RIGHT       >      "MODE" AND "RB" DON'T APPLY, RA=0 ILLEGAL
                                ;       110700  SWAP BYTES (RA)        /
                                ;
                                ;       BRANCH INSTRUCTIONS             OP + ADDRESS
                                ;
                                ;       OP
                                ;
                                ;       0       JUMP                            JUMP 0 IS RETURN
                                ;       10000   JUMP IF ZERO
                                ;       20000   JSB (JUMP SUBROUTINE)
                                ;       30000   JUMP IF POSITIVE
                                ;       40000   JUMP IF CARRY CLEAR
                                ;       50000   JUMP IF NON-ZERO
                                ;       60000   XFC (ESCAPE TO MICROCODE) - XFC 0 IS BREAKPOINT
                                ;       70000   JUMP IF NEGATIVE
                                ;
                                ;       REGISTER USAGE:
                                ;
                                ;       RPC     (R12)   DM PROGRAM COUNTER
                                ;       RIB     (R13)   HOLDS CURRENT DM INSTRUCTION
                                ;       RCC     (R14)   HOLDS LAST ARITHMETIC/LOGICAL RESULT FOR COND CODES
                                ;       RCY     (R15)   HOLDS CARRY OF LAST ARITHMETIC OP
                                ;       RT1     (R16)   GENERAL PURPOSE TEMP #1
                                ;       RT2     (R17)   GENERAL PURPOSE TEMP #2
                                ;
```

```
                                ;       DM REGISTERS ARE STORED IN FIRST 8 LOCATIONS OF BUFFER MEMORY
                                ;       IN GENERAL, DM RN = LOCATION N; EXCEPTIONS: R0=LOC 7, SP = LOC 6
                                ;;-
```



```
                                DMNTRY:         ;ENTER DM HERE TO ZAP HIGH-ORDER CARRY BITS
003626  033752  000365  010000        MOV     #DM.BEG+DMSTR&LOBYT,RPC         ; [16K]RPC=LO DM START ADDR
003627  133552  000002  000000        BIS     #DM.BEG&HIBYT,RPC               ; [16K]RPC=HI DM START ADDR

003630  034455  020015  013734  GETCRY: CLR     RCY             %JNCRY  SETCRY  ; SET CARRY (INVERTED) FROM LAST OP

                                ;       FETCH IS THE BEGINNING OF EVERY DM CYCLE

003631  120452  007012  000000  FETCH:  INC     RPC\0,RPC,BAR                   ; START FETCH AND BUMP PC
                                DMPC:   ;TRACE THIS ADDRESS FOR TO SEE DM PROGRAM COUNTER
003632  114544  000040  010000  FETCH2: BIT     #DMBEG,RLL                      ; IF DM NOT RUNNING
003633  033713  010003  143776        MOV\F   (BUF),RIB       %JZRO   OXFC17  ;   THEN DONE/ELSE GET DM INSTRUCTION
003634  114553  030020  013752  FETCH3: BIT     #10000,RIB      %JNNEG  BRANCH  ; BRANCH CLASS IS POSITIVE
003635  014553  010200  013647        BIT     #200,RIB        %JNZRO  SNGLOP  ; 1-OP INSTS HAVE 10000 ON
003636  114553  010001  103706        BIT     #400,RIB        %JZRO   MODE04  ; DECODE THE SRC/DST MODE
003637  053456  010013  013676        MOV\R   RIB,RT1         %JNZRO  MODE6   ; START SHIFTING RB INTO <2:0>

                                ;       AUTOINCREMENT MODE - CHECK FOR RB=0 (IMMEDIATE OR ABSOLUTE MODE)

003640  054556  000034  010000        AND\R   #7*4,RT1                        ; ISOLATE RB
003641  014553  010100  113645        BIT     #100,RIB        %JZRO   IMMED   ; BR IF IMMED/ABS MODE AND SEPARATE THEM
003642  053456  000016  123703        MOV\R   RT1,RT1         %CALL   GETRBX  ; NOT IMMED/ABS MODE, FETCH REGISTER
003643  130454  000016  010000        INC     RT1,RCC                         ; MAKE INCREMENTED COPY
003644  013440  003014  113713        MOV     RCC,BUF         %JMP    CKSTO   ; STORE COPY, SRC ADR = ORIGINAL
                                                                                ; CODE SHARED WITH AUTODECREMENT

003645  120452  017012  103716  IMMED:  INC     RPC\0,RPC,BAR   %JZRO   OPDCD   ; BR IF IMMED MODE - SOURCE = NEXT WORD
003646  031456  000012  103713        DEC     RPC,RT1         %JMP    CKSTO   ; ABS MODE - ALLOW STORES
```

```
                                ;       SINGLE-OPERAND INSTRUCTIONS COME HERE FOR DECODE

003647  114553  010001  013652  SNGLOP: BIT     #400,RIB        %JNZRO  ROTATE  ; ROTATES HAVE 200 BIT SET
003650  033756  010000  113717          MOV     #0,RT1          %JZRO   OPDCD1  ; SOME OPS TAKE AN IMPLIED 0
003651  033756  000001  113717          MOV     #1,RT1          %JMP    OPDCD1  ; WHILE SOME TAKE AN IMPLIED 1

003652  014553  007007  010000  ROTATE: AND     #7,RIB\N,BAR                    ; START REG FETCH, DECODE DIRECTION
003653  120472  017012  163750          INC\T   RPC\0,RPC,BAR   %CZRO   INDRCT  ; IF MEMORY THEN GET ADDR
003654  033714  000003  000000          MOV     (BUF),RCC                       ; GET REGISTER/MEM IN RCC
003655  114553  000001  010000          BIT     #400,RIB                        ; CHECK FOR ROR
003656  014553  010100  003662          BIT     #100,RIB        %JNZRO  ROR     ; IF ROR THEN CONTINUE
003657  030154  000014  010000  ROL:    ADD     RCC,RCC                         ; DOUBLE THE REGISTER
003660  033154  023015  103734          BIS     RCY,RCC,BUF     %JCRY   SETCRY  ; LSB=OLD CARRY, BR ON NEW CARRY
003661  034455  000015  103631          CLR     RCY             %JMP    FETCH   ; SET NEW CARRY TO 0

003662  016155  010014  003666  ROR:    XOR     RCC,RCY\N       %JNZRO  SWAB    ; SEE IF NEW CARRY SAME AS OLD
003663  053454  040014  103673          MOV\R   RCC,RCC         %JNLSB  STORCC  ; IF SO, 16-BIT ROT = 17 BIT ROT
003664  136554  000200  000000          XOR     #100000,RCC                     ; OTHERWISE MUST CHANGE NEW SIGN
003665  036555  000001  103673          XOR     #1,RCY          %JMP    STORCC  ; AND NEW CARRY

003666  053454  000014  133674  SWAB:   MOV\R   RCC,RCC         %CALL   ROTR3   ; SHIFT 4 BITS
003667  053454  000014  133674          MOV\R   RCC,RCC         %CALL   ROTR3   ; AND ANOTHER 4 BITS
003670  053454  003014  133674          MOV\R   RCC,RCC,BUF     %CALL   ROTR3   ; SHIFT BACK 4 BITS
003671  053454  000014  133674          MOV\R   RCC,RCC         %CALL   ROTR3   ; AND ANOTHER 4 BITS
003672  134554  000377  103631          AND     #HIBYT,RCC      %JMP    FETCH   ; ISOLATE THE CC/GO FETCH NEXT INST
003673  013440  003014  103631  STORCC: MOV     RCC,BUF         %JMP    FETCH   ; STORE ANSWER AND RETURN

003674  053454  000014  123675  ROTR3:  MOV\R   RCC,RCC         %CALL   ROTR1   ; ROTATE RIGHT 1 AND ANOTHER
003675  053454  000014  137777  ROTR1:  MOV\R   RCC,RCC         %RET            ; ROTATE RIGHT 1 AND RETURN
```



```
                                ;       SRC/DST MODE IS MEMORY REFERENCE - ADDRESS OF OPERAND IS NEXT WORD
                                ;       INDEXED BY REGISTER SPECIFIED IN INSTRUCTION

003676  054556  000034  010000  MODE6:  AND\R   #7*4,RT1                        ; ISOLATE RB (COMPLEMENTED)
003677  120452  017012  103704          INC     RPC\0,RPC,BAR   %JZRO   PCREL   ; START DISPLACEMENT FETCH
                                                                                ; AND BREAK OUT MODE 60
003700  033714  000003  123702          MOV     (BUF),RCC       %CALL   GETRB   ; GET DISPLACEMENT IN RCC, AND
                                                                                ; FETCH INDEX REGISTER INTO RT1
003701  030156  000014  113713          ADD     RCC,RT1         %JMP    CKSTO   ; FORM FINAL ADDR, CHECK IF STORE

003702  053456  000016  010000  GETRB:  MOV\R   RT1,RT1                         ; GET RB INTO <2:0> OF RT1
003703  013440  007016  103610  GETRBX: MOV     RT1,BAR         %JMP    LDRT1   ; ROUTINE TO SET RT1 = (RT1)

                                ;       IF RB=0 AND MODE=6, ADDRESSING MODE IS PC RELATIVE

003704  033716  000003  010000  PCREL:  MOV     (BUF),RT1                       ; GET DISPLACEMENT WORD
003705  030156  000012  113713          ADD     RPC,RT1         %JMP    CKSTO   ; USE PC INSTEAD OF INDEX REG

                                ;       COME HERE IF SRC/DST MODE IS REG OR -(REG)

003706  053456  010013  013711  MODE04: MOV\R   RIB,RT1         %JNZRO  MODE4   ; BREAK OUT THE TWO MODES
003707  054556  000034  010000          AND\R   #7*4,RT1                        ; ISOLATE RB, SHIFT IT INTO <2:0>
003710  053456  000016  103713          MOV\R   RT1,RT1         %JMP    CKSTO   ; CHECK FOR STORE,USE RB AS SRC

                                ;       AUTODECREMENT REGISTER MODE - DECREMENT RB, OPERAND ADDRESS IS NEW RB

003711  054556  000034  133702  MODE4:  AND\R   #7*4,RT1         %CALL   GETRB   ; ISOLATE RB, SHIFT IT INTO <2:0>
                                                                                ; AND FETCH ITS CONTENTS
003712  031456  000016  123621          DEC     RT1             %CALL   STORT1  ; DECREMENT CONTENTS OF REGISTER
003713  114553  000016  010000  CKSTO:  BIT     #7000,RIB                       ; TEST FOR STORE INSTRUCTION
003714  014553  010100  113744          BIT     #100,RIB        %JZRO   STORE   ; BR IF STORE, TEST INDIRECTION
003715  013440  017016  033750          MOV     RT1,BAR         %CNZRO  INDRCT  ; PUT DIRECT/INDIRECT ADDR IN BAR
```

```
                                ;       MAIN OPERATION DECODE AND EXECUTION - SRC ADDR IN BAR

003716  033716  000003  010000  OPDCD:  MOV     (BUF),RT1                       ; GET SOURCE OPERAND
003717  014553  007007  010000  OPDCD1: AND     #7,RIB\N,BAR                    ; START FETCH OF DEST REG
003720  114553  010002  103742  OPDCD2: BIT     #1000,RIB       %JZRO   MEMDST  ; RA=0 MEANS DEST IS MEMORY!
003721  114553  010004  013727          BIT     #2000,RIB       %JNZRO  OPXX1   ; OP CODE IS 3 BITS WIDE
003722  114553  010010  013724  OPXX0:  BIT     #4000,RIB       %JNZRO  OPX10   ; SO IT TAKES 3 LEVELS OF DECODE
003723  033454  003016  103631  OLOAD:  MOV     RT1,RCC,BUF     %JMP    FETCH   ; LOAD INSTRUCTION IS SIMPLE MOVE

003724  033714  010003  103726  OPX10:  MOV     (BUF),RCC       %JZRO   OBTST   ; FETCH DEST REG FOR TEST INSTS
003725  132154  000016  113630  OCOMP:  RSUB    RT1,RCC         %JMP    GETCRY  ; COMPARE INST SETS COND CODES
003726  034154  000016  113631  OBTST:  AND     RT1,RCC         %JMP    FETCH   ; SO DOES BIT TEST INST

003727  114553  010010  013735  OPXX1:  BIT     #4000,RIB       %JNZRO  OPX11   ; CONTINUE DECODING
003730  033714  010003  013732  OPX01:  MOV     (BUF),RCC       %JNZRO  OTADD   ; FETCH DEST REG AND DO FINAL DCD
003731  033154  003016  103631  OBSET:  BIS     RT1,RCC,BUF     %JMP    FETCH   ; BIT SET INST DOES WHAT IT SAYS
003732  030154  000016  000000  OTADD:  ADD     RT1,RCC                         ; ADD INST TOO SLOW TO STORE RSLT
003733  024455  023014  003631          CLR     RCC\0,RCY,BUF   %JNCRY  FETCH   ; STORE RSLT, CLR CARRY & TEST IT
003734  033755  000001  113631  SETCRY: MOV     #1,RCY          %JMP    FETCH   ; SET DM CARRY IF 2901 CARRY SET

003735  033714  010003  013737  OPX11:  MOV     (BUF),RCC       %JNZRO  OSUBT   ; FETCH DEST REG AND DO FINAL DCD
003736  035154  003016  103631  OBCLR:  BIC     RT1,RCC,BUF     %JMP    FETCH   ; BIT CLEAR INST DOES THE OBVIOUS
003737  131154  000016  000000  OSUBT:  SUB     RT1,RCC                         ; SUB INST TOO SLOW TO STORE RSLT
003740  024455  023014  113631          CLR     RCC\0,RCY,BUF   %JCRY   FETCH   ; STORE RESULT AND COMPUTE COMP-
003741  033755  000001  113631          MOV     #1,RCY          %JMP    FETCH   ; LEMENT OF 2901 CARRY AS C-BIT

                                ;       COME HERE FROM OPDCD2 IF RA = 0, MEANING MEMORY DEST (A LA PDP-11 MODE 37)

003742  120452  007012  123606  MEMDST: INC     RPC\0,RPC,BAR   %CALL   S.LDCC  ; GET MEMORY ADDR FROM INST STREAM
003743  133554  007200  103720          BIS     #100000,RCC,BAR %JMP    OPDCD2  ; FORCE NON-ZERO AND RE-DECODE OP

                                ;       ALL STORES COME HERE (EXCEPT MODE 27O WHICH IS ILLEGAL)
                                ;       WITH TARGET ADDRESS (LESS INDIRECTION) IN RT1

003744  014553  007007  133606  STORE:  AND     #7,RIB\N,BAR    %CALL   S.LDCC  ; START FETCHING SOURCE REGISTER
003745  014553  000100  000000          BIT     #100,RIB                        ; TEST FOR INDIRECTION
003746  013440  017016  033750          MOV     RT1,BAR         %CNZRO  INDRCT  ; SET UP ADDRESS TO STORE IT IN
                                ; ***   VALIDATE RAM ADDRESS TO BE STORED INTO, 0 -> 7, 556 -> 7777 LEGAL ***
                                ;       BIC     #170000,RT1                     ; ISOLATE ADDRESS (12 BITS)
                                ;       CMP     #8.,RT1                         ; IF LESS THAN 8
                                ;       ASSUME  DM.BEG&1,EQ,0                   ; MAKE SURE DM.BEG IS EVEN
                                ;       MOV\LF  #<DM.BEG/2>,RIB %JNCRY  STORB   ;   THEN OK/ELSE RIB=556.
                                ;       CMP     RIB,RT1                         ; IF RT1 >= 556
                                ;       MOV\F   #ER.DMI,R11     %JCRY   STORB   ; THEN DO STORE/ELSE CONTROLLE
                                                                                ; FATAL ERROR
                                ;STORA: NOP                                     ; SO D.PROC WON'T TIMEOUT
                                ;       BIS     #CN.ERR,RLL     %JMP    STORA   ; AND HANG IT UP HERE
003747  013440  003014  103631  STORB:  MOV     RCC,BUF         %JMP    FETCH   ; STORE REG IN DEST AND GET OUT

003750  033716  000003  010000  INDRCT: MOV     (BUF),RT1                       ; RT1=INDIRECT PTR
003751  013440  007016  127777          MOV     RT1,BAR         %RET            ; BAR=INDIRECT PTR
```



```
                                ;       BRANCH CLASS INSTRUCTIONS COME HERE FROM "FETCH" FOR EXECUTION

003752  120452  007012  123611  BRANCH: INC     RPC\0,RPC,BAR   %CALL   S.LD17  ; [16K]R17=BRANCH ADDR
003753  114553  000020  010000          BIT     #10000,RIB                      ; [16K]DECODE BRANCH TYPE
003754  114553  010040  014011          BIT     #20000,RIB      %JNZRO  JCOND   ; DECODE 3 BIT BRANCH TYPE
003755  114553  017100  003765          AND     #40000,RIB\N,BAR %JNZRO JSBXFC  ; [16K]TEST RIB FOR COND/XFC
003756  033452  010017  053764          MOV\F   R17,RPC         %JNZRO  OJCC    ; [16K]DO UNCONDITIONAL BR HERE
003757  033753  017006  013631          MOV     #6,RIB,BAR      %JNZRO  FETCH   ; TEST FOR RETURN INST

                                ;       RETURN IS ENCODED AS AN CONDITIONAL JUMP ADDRESS OF 0

003760  033716  000003  010000  RETURN: MOV     (BUF),RT1                       ; GET VALUE OF STACK PTR
003761  013440  007016  133604          MOV     RT1,BAR         %CALL   S.LDPC  ; SET BAR/LOAD PC
003762  120356  007013  133621          INCB    RIB\0,RT1,BAR   %CALL   STORT1  ; INCR STACK PTR & GO STORE IT
003763  120452  007012  103632          INC     RPC\0,RPC,BAR   %JMP    FETCH2  ; START FETCH OF NEXT INST

003764  013455  000015  114013  OJCC:   TST     RCY             %JMP    JEQJPL  ; JUMP ON CARRY CLEAR - USE JEQ

003765  031472  010012  043772  JSBXFC: DEC\T   RPC,RPC         %JNZRO  OXFC    ; [U52EC1]BREAK JSB OUT FROM XFC
003766  013740  007006  123610          MOV     #6,BAR          %CALL   LDRT1   ; FETCH STACK PTR
003767  031456  000016  123621          DEC     RT1             %CALL   STORT1  ; DECREMENT STACK PTR & STORE IT
003770  013440  007016  000000          MOV     RT1,BAR                         ; POINT BAR AT NEW TOP OF STACK
003771  135552  003300  104016          BIC     #140000,RPC,BUF %JMP    JNEJMI  ; [16K]SAVE PC AND START NEXT FETCH
                                ; ***   XFC PROCESSING ***
003772  033756  000354  010000  OXFC:   MOV     #DM.BEG+DMBPC&LOBYT,RT1         ; [16K]RT1=LO BYTE OF ODT PC
003773  133556  007002  133615          BIS     #DM.BEG&HIBYT,RT1,BAR %CALL S.STPC; [16K]SAVE PC FOR ALL XFC'S
003774  135553  000140  000000          BIC     #60000,RIB                      ; CLEAR EXTRANEOUS BITS FROM CODE
003775  011553  010022  104003          SUBC    #XFCMAX,RIB\N   %JZRO   OXFCO   ; [16K]IF XFC 0 THEN BKPT/ELSE IF XFC ILLEGAL
003776  033773  030021  053777  OXFC17: MOV\T   #XFCFIN,RIB     %TNMSB          ;   THEN FORCE XFC STOP !
003777  011553  000014  010000          SUBC    #XFCPAS,RIB\N                   ; IF XFC PROCESSED BY U.PROC
004000  033471  030013  053561          MOV\T   RIB,R11         %JNMSB  XF11A   ;   THEN INVOKE U.PROC
004001  030553  000343  000000          ADD     #<XFCTAB-1>&LOBYT,RIB           ;   ELSE RIB EQ LO XFCTAB ADDR
004002  133553  002377  010000          OR      #HIBYT,RIB,PAR                  ; CRAM PAR WITH XFC ADDRESS
                                ; ***   DM BREAKPOINT PROCESSING ***
004003                                  ASSUME  RCC,EQ,R14                      ; [16K]MAKE SURE RCC==R14
004003                                  ASSUME  RCY,EQ,R15                      ; [16K]MAKE SURE RCY==R15
004003                          OXFCO:  ASSUME  <DMBEG+ODTAF&17>,EQ,<<DMBEG+DMBPC&17>+1>
004003                                  ASSUME  <DMBEG+ODTCY&17>,EQ,<<DMBEG+ODTAF&17>+1>
004003  013440  003014  133620          MOV     R14,BUF         %CALL   S.ST15  ; [U52EC1][16K]SAVE CARRY FLAG
004004  030556  007010  123610          ADD     #<DMODT-DMBPC>,RT1,BAR %CALL LDRT1; [16K]RT1=ODT START ADDR
004005  033472  010016  054010          MOV\T   RT1,RPC         %JNZRO  XFCRTA  ; [16K]IF NEQ 0 THEN START IT
004006  033756  000354  010000  XFCRET: MOV     #DM.BEG+DMBPC&LOBYT,RT1         ; [16K]RT1=LO BYTE OF DM PC
004007  133556  007002  133604          BIS     #DM.BEG&HIBYT,RT1,BAR %CALL S.LDPC; [16K]RESTORE DM PC FROM MEMORY
004010  034555  000001  113631  XFCRTA: AND     #1,RCY          %JMP    FETCH   ; MAKE SURE CARRY IS LEGAL

                                ;       CONDITIONAL JUMP INSTRUCTIONS (JEQ, JNE, JPL, JMI)

004011  114553  010100  004015  JCOND:  BIT     #40000,RIB      %JNZRO  JSIGN   ; BREAK OUT SIGN JMPS FROM 0 JMPS
004012  013454  010014  014016          TST     RCC             %JNZRO  JNEJMI  ; BREAK OUT JEQ FROM JNE
004013  120472  017017  153632  JEQJPL: INC\T   R17\0,RPC,BAR   %JZRO   FETCH2  ; [16K]SET NEW PC AND FETCH IF 0
004014  120452  007012  103632          INC     RPC\0,RPC,BAR   %JMP    FETCH2  ; USE OLD PC AND FETCH IF NOT

004015  114554  010200  104013  JSIGN:  BIT     #100000,RCC     %JZRO   JEQJPL  ; BREAK OUT JPL FROM JMI
004016  120472  017017  043632  JNEJMI: INC\T   R17\0,RPC,BAR   %JNZRO  FETCH2  ; [16K]SET NEW PC & FETCH IF NON-0
```

```
004017  120452  007012  103632          INC     RPC\0,RPC,BAR   %JMP    FETCH2  ; USE OLD PC AND FETCH IF NOT
```



```
                                ;       DIAGNOSTIC MACHINE DEBUGGING TECHNIQUE
                                ;
                                ;       EXAMINES AND DEPOSITS REGISTERS AND MEMORY AND SETS BREAKPOINTS
                                ;       FOR THE DIAGNOSTIC MACHINE, USING A DL-11 ON THE UNIBUS.
                                ;
                                ;
                                ;       FLAG DEFINITIONS:

000001                          NUMFLG  =       1               ; NUMBER SEEN
000002                          OPNFLG  =       2               ; LOCATION OPEN
000010                          ZFLG    =       10              ; SIGNIFICANCE FLAG FOR NUMERIC OUTPUT

177560                          DL11AD  =       177560          ; ADDRESS OF CONSOLE DL11
176510                          DL1144  =       176510          ; ADDRESS OF 11/44 CONSOLE
                                ;       WORK LOCATIONS USED BY DMDT

                                ;QREG   =       INPUT           ; STORAGE FOR 3 EASY-TO-RECALL NUMBERS + ...

                                ;       REGISTER DEFINITIONS

000017                          RBPA    =       R17             ; UNUSED LOWER-MICRO REGISTER WHEN DM RUNNING


                                ;       ENTER HERE ON BREAKPOINT OR INITIALLY

                                DMDT:
                                ;                                               ; [16K]NO DMODT !!!!!
```

```
                                .SBTTL  HARDWARE/SOFTWARE VECTOR TABLES

                                .ORG    ^HOF00
007400                          .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO ERROR ROUTINE

                                .ORG    ^HOF03
                                ;;*****************************************************************************
                                ;
                                ;       THE PAR (Pipeline Address Register) IS FORCED HERE TO THE LO VECTOR
                                ;       TABLE AND TO THE HI VECTOR TABLE DURING THE PAR TEST.
                                ;
                                ;       NOTE:   THE LOCATION LOPAR (^HOF03) AND HIPAR (^HOFFC), WHICH ARE USED
                                ;               IN THE PAR TEST, MUST NOT BE CHANGED.
                                ;
                                ;;*****************************************************************************

007403  013440  000000  117774  LOPAR:  NOP                     %JMP    HIPAR   ;JUMP TO HIPAR

                                ;;*****************************************************************************
                                ;
                                ; TABLE NAME:
                                ;       DTSTBL
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       This table contains the addresses of D-PROC tests that can be looped
                                ;       on.
                                ;
                                ;;*****************************************************************************

000004                          TESTBL  = . - 7400

007404  034451  000011  110053  DTSTBL: CLR     DER             %JMP    HD.T0   ;HANG D-PROC, LET U-PROC LOOP
007405  034451  000011  110053          CLR     DER             %JMP    HD.T1   ;HANG D-PROC, LET U-PROC LOOP
007406  034451  000011  110053          CLR     DER             %JMP    HD.T2   ;HANG D-PROC, LET U-PROC LOOP
007407  034451  000011  100054          CLR     DER             %JMP    T3      ;LOOP D-PROC TEST
007410  034451  000011  110060          CLR     DER             %JMP    T4      ;LOOP D-PROC TEST
007411  034451  000011  100064          CLR     DER             %JMP    T5      ;LOOP D-PROC TEST
007412  034451  000011  100067          CLR     DER             %JMP    T6      ;LOOP D-PROC TEST
007413  034451  000011  100075          CLR     DER             %JMP    T7      ;LOOP D-PROC TEST
007414  034451  000011  100100          CLR     DER             %JMP    HD.T8   ;HANG D-PROC, LET U-PROC LOOP
007415  034451  000011  110101          CLR     DER             %JMP    T9      ;LOOP D-PROC TEST
```



```
                                ;++
                                ; TABLE NAME:
                                ;       VECTAB  (CONTROLLER HARDWARE ERROR, STATE AND ERROR VECTOR TABLE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS TABLE CONTAINS ALL OF THE VECTOR TABLES FOR BOTH THE U.PROC
                                ; AND D.PROC. CONTROLLER MICROCODE LOADING THE 'PAR' REGISTER IN PREPARATION FOR
                                ; ENTERING THIS TABLE NEED ONLY LOAD THE LEAST SIGNIFICANT 8 BITS (LOW BYTE)
                                ; OF THE REQUIRED OFFSET, THE CONTROLLER HARDWARE WILL FORCE THE MOST SIGNIFICANT
                                ; 8 BITS (HI BYTE) TO ALL ONES. THE CONTROLLER ERROR DETECTING HARDWARE USES THE
                                ; LAST 16 ENTRIES OF THE VECTOR TABLE AND THESE ENTRIES MUST START AT ADDRESS
                                ; OFF0 HEX). A DESCRIPTION OF HOW THE HARDWARE FORMS THE VECTOR ADDRESS
                                ; AND THE ERRORS DETECTED FOLLOW:
                                ;
                                ;       ERROR VECTOR ADDRESS - GENERAL
                                ;               THE ERROR VECTOR ADDRESS IS 'BIT ENCODED' BASED ON THE ERROR
                                ;               WHICH OCCURRED.   IN ALL CASES, THE INSTRUCTION IN WHICH THE
                                ;               ERROR OCCURRED WILL COMPLETE, THE FOLLOWING INSTRUCTION WILL
                                ;               COMPLETE, AND THE NEXT INSTRUCTION WILL BE THE ERROR VECTOR.
                                ;
                                ;               SEQ ADR BIT 03 OCTAL DECODE;
                                ;                       0= D PROCESSOR
                                ;                       1= U PROCESSOR
                                ;               SEQ ADR BITS 02-00 OCTAL DECODE;
                                ;                       0= NOT USED BY HARDWARE
                                ;                       1= BI READ PARITY ERROR 2
                                ;                       2= BI WRITE PARITY ERROR 2
                                ;                       3= BCI PARITY ERROR 1
                                ;                       4= NOT USED BY HARDWARE
                                ;                       5= RAM PARITY ERROR
                                ;                       6= TIME OUT ERROR
                                ;                       7= CROM PARITY ERROR
                                ;
                                ;       CONTROL MEMORY PARITY ERROR
                                ;               THE  PROCESSOR  WITH THE  CROM PARITY ERROR  WILL  EXECUTE  THE
                                ;               FAULTY INSTRUCTION, THE FOLLOWING INSTRUCTION, THEN  THE  ERROR
                                ;               VECTOR INSTRUCTION.  THE SIGNAL 'CROM PE L'  WILL BECOME ACTIVE
                                ;               ON THE CYCLE FOLLOWING THE ERROR. THIS CORRESPONDS TO THE THIRD
                                ;               INSTRUCTION AFTER THE ERROR WHEN USING THE LOGIC ANALYSER.
                                ;               IF THE VECTOR ADDRESS HAS A PARITY ERROR, THE PROCESSOR
                                ;               WILL HANG. (UNTIL IT GOES AWAY)  SO DISPLAY STATUS.
                                ;
                                ;       TIMEOUT ERROR
                                ;               THE TIMEOUT CIRCUIT IS ENABLED WHEN EVER A SUCCESSFUL JUMP IS
                                ;               EXECUTED. FALLING THROUGH TO THE NEXT INSTRUCTION RESETS THE
                                ;               TIMEOUT CIRCUIT. NOTE - TIMEOUT ERRORS DO NOT OCCUR ON THE
                                ;               UNIBUS PROCESSOR.
                                ;
                                ; ***   THE HARDWARE MUST BE CHANGED TO DISABLE THE UNIBUS TIMEOUT.*****
                                ;
                                ;TIME=  <16600.*7/1000.>                ; MSEC MINIMUM TIMEOUT
                                ;TIME=  <16600.*8/1000.>                ; MSEC MAXIMUM TIMEOUT
                                ;INST=  <16600.*7*1000./INSTR>          ; INSTR MINIMUN
```

```
                                ;--
                                ; ***   NULL AREA WITH JUMP TO ERROR ROUTINE (+PARITY) - JUST IN CASE ***

007416                                  .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO ERROR ROUTINE

                                ; ***   NOTE - IF VECTAB SIZE CHANGES THEN THE PRECEEDING .ORG MUST ALSO ***
                                ; ***   THE EQUATION IS .ORG VALUE = FE9H - SIZE OF VECTAB ***

                                .ORG    ^HOFA0                                  ; [mjt08]ORIGIN OFA0 HEX
007640  133544  010010  087777 D.STPA: BIS\F   #CN.ERR,RLL      %RNZRO          ; [U52EC1][16K]IF ZERO, EXIT/CONTROLLER ERR BIT IN RLL
007641  013740  007034  123614         MOV     #FAILUR,BAR     %CALL   S.ST11  ; [U52EC1]SAVE LAST FAIL CODE IN CASE U.PROC DOESN'T
007642  114544  000002  107640 D.STOP: BIT     #INDIAG,RLL     %JMP    D.STPA  ; [U52EC1][16K]CHECK IF DIAGNOSTIC

                                ; ***   START CONTROLLER VECTOR TABLE ***
                                ; ***   REVECTORING STATE VECTORS - MUST START ON ODD WORD.
                                ;       CODE IN VECTOR IS TO STORE INCREMENTED VECTOR PTR BACK IN MEMORY
007643                          VECTAB: ASSUME  .&1,EQ,1                        ; UPROC STARTS FIRST.....
007643  013440  003017  113273 V.DRVC: MOV     R17,BUF         %JMP    D.RVC2  ; INIT RVCBUF, COMPUTE INITIAL RCT OFFSETS
007644  033717  000003  103323 V.RVC3: MOV     (BUF),R17       %JMP    D.RVC4  ; COMPLETE READ OF RCT BLOCK
007645  033717  000003  103323 V.RVC5: MOV     (BUF),R17       %JMP    D.RVC4  ; DO OPERATION ON TARGET BLOCK
007646  033717  000003  113342 V.RVC8: MOV     (BUF),R17       %JMP    D.RVC8  ; CLEAN UP AND EXIT
007647  033717  000003  113342         MOV     (BUF),R17       %JMP    D.RVC8  ; [mjt08]CLEAN UP AND EXIT
007650  033717  000003  113342         MOV     (BUF),R17       %JMP    D.RVC8  ; [mjt08]CLEAN UP AND EXIT

                                ; ***   TOPOLOGY PROCESSING STATE VECTORS ***

007651  033752  000220  111341 V.TOPO: MOV     #TOPOL,R12      %JMP    LEV2.1  ; SEND TOPOLOGY COMMAND
007652  033752  000157  111445         MOV     #TOPRSP,R12     %JMP    R.TOPO  ; RECEIVE TOPOLOGY RESPONSE
007653  033751  000004  112037         MOV     #<V.DCLR-.>,R11 %JMP    D.INSV  ; LINK INTO DRIVE CLEAR FLOW

                                ; ***   LEVEL 0 ERROR AND I/O ERROR STATE VECTORS ***

007654  013440  000000  111465 V.INIT: NOP                         %JMP    S.INIT  ; SEND DRIVE INIT
                                V.ATTN:                                         ; ATTENTION'S USE THIS TOO
007655  033752  000011  111341 V.ERRO: MOV     #GETSTA,R12     %JMP    LEV2.1  ; SEND GET STATUS COMMAND
007656  033752  000366  111445         MOV     #STARSP,R12     %JMP    R.GSTA  ; RECEIVE GET STATUS RESPONSE
007657  034556  000357  111401 V.DCLR: AND     #<LOBYT-DRV.DF>,R16 %JMP S.DCLR ; SEND DRIVE CLEAR COMMAND
007660  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE DRIVE CLEAR RESPONSE
007661  033751  000031  101773 V.ERRC: MOV     #<V.ATT2-V.ERRC>,R11 %JMP D.ERRO; DECIDE WHAT TO DO NEXT
007662  033752  000216  111506         MOV     #RECALB,R12     %JMP    S.LRSP  ; GO SEND RECALIBRATE COMMAND
007663  033753  000011  101476         MOV     #INPUT,R13      %JMP    R.LRSP  ; RECEIVE LONG TIMEOUT RESPONSE
007664  033751  000016  112037 V.ERSK: MOV     #<V.SEEK-.>,R11 %JMP    D.INSV  ; LINK INTO SEND SEEK FLOW

                                ; ***   SEEK PROCESSING STATE VECTORS ***

007665  033757  000017  111467 V.SKON: MOV     #SDITMO,R17     %JMP    S.ONLN  ; SEND ONLINE COMMAND
007666  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE DRIVE CLEAR RESPONSE
007667  033752  000011  111341         MOV     #GETSTA,R12     %JMP    LEV2.1  ; SEND GET STATUS COMMAND
007670  033752  000366  111445         MOV     #STARSP,R12     %JMP    R.GSTA  ; RECEIVE GET STATUS RESPONSE
007671  135557  003014  101504         BIC     #<DRV.AV!DRV.AT>,R17,BUF %JMP S.RUNN ; [16K]SEND RUN COMMAND
007672  033753  000011  101476         MOV     #INPUT,R13      %JMP    R.LRSP  ; RECEIVE LONG TIMEOUT RESPONSE
007673  033752  000210  101441         MOV     #GTUCHR,R12     %JMP    S.GUCH  ; [U52EC1]SEND GET SUBUNIT CHAR COMMAND
```



```
007674  033754  000016  101443         MOV     #<SDI.LL-SDI.H1>,R14 %JMP R.GUCH ; [U52EC1]RECEIVE SUBUNIT CHAR
007675  133753  000201  111360 V.CMD1: MOV     ##CHGMOD,R13    %JMP    S.CMOD  ; [V05]SEND STYLE 1 CHANGE MODE COMMAND
007676  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; [V05]RECEIVE CHANGE MODE RESPONSE
007677  013440  000600  111367         NOP             @RDPF   %JMP    S.CFL1  ; [V05]SEND STYLE 1 CHANGE FLAG
007700  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; [V05]RECEIVE CHANGE FLAG RESPONSE
007701  013440  000000  111613         NOP                     %JMP    C.IMFF  ; [V05]CHECK IF IGNORE MEDIA FORMAT FLAG SET
007702  033456  000015  101510 V.SEEK: MOV     DBAR,R16        %JMP    S.SEEK  ; SEND SEEK COMMAND
007703  033753  000011  101536         MOV     #INPUT,R13      %JMP    R.SEEK  ; RECEIVE SEEK RESPONSE
007704  033753  000002  101673         MOV     #<.-V.SEEK>,R13 %JMP    C.SEEK  ; SEEK COMPLETE CHECK
                                                                                ; R13=ERROR RECOVERY BACKUP VALUE
                                ; ***   LEVEL 1 ERROR STATE VECTORS ***

007705  033752  000006  111426 V.ERR1: MOV     #ERECOV,R12     %JMP    S.EREC  ; SEND ERROR RECOVERY COMMAND
007706  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE ERROR RECOVERY RESPONSE
007707  033752  000001  102022         MOV     #PKIP,R12       %JMP    D.DONE  ; STATE VECTOR COMPLETION ROUTINE
                                                                                ; R12=PKIP FOR INOCUOUS NONZERO VALUE
007710  013440  000000  107710 V.NULL: NOP                     %JMP    V.NULL  ; [V05]NULL ENTRY

                                ; ***   ONLINE COMMAND STATE VECTORS ***

                                V.ONLN:
007711  013440  000000  102071 V.OLRD: NOP                     %JMP    D.OLRD  ; GO DO ONLINE READ OF FCT

                                ; ***   ATTENTION CONTINUATION VECTORS ***

007712  010555  007044  101471 V.ATT2: ADD     #SDI.S1,DBAR\N,BAR %JMP S.RCAL  ; SEND RECALIBRATE COMMAND
007713  033753  000011  101476         MOV     #INPUT,R13      %JMP    R.LRSP  ; RECEIVE LONG TIMEOUT RESPONSE
007714  133753  000201  111360 V.SUCH: MOV     ##CHGMOD,R13    %JMP    S.CMOD  ; [V05]SEND STYLE 2 CHANGE MODE COMMAND
007715  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE CHANGE MODE RESPONSE
007716  037457  000012  101665         COM     R12,R17         %JMP    C.DCON  ; CHECK FOR NEED TO SEND DISCONNECT COMMAND
007717  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE DISCONNECT RESPONSE
007720  013440  000000  102074         NOP                     %JMP    D.ZATT  ; STATE VECTOR COMPLETION ROUTINE

                                ; ***   AVAILABLE SPIN DOWN STATE VECTORS ***

007721  013440  000400  101367 V.AVAL: NOP             @SDPF   %JMP    S.CFL1  ; SEND STYLE 2 CHANGE CONTROLLER FLAGS
007722  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE CHANGE CONTROLLER FLAGS RESPONSE

                                ; ***   AVAILABLE AND SPIN DOWN AFTER ONLINE ERROR ***

007723  133557  003010  102036         BIS     #DRV.AV,R17,BUF %JMP    D.INS1  ; [U52EC2]SET AVAILABLE AND STEP TO NEXT VECTOR
007724  033456  000011  111414 V.ONAV: MOV     R11,R16         %JMP    S.DCN2  ; [U52EC2]SEND DISCONNECT COMMAND
007725  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE DISCONNECT RESPONSE
007726  037151  000011  112073         XNOR    R11,R11         %JMP    D.WAIT  ; STATE VECTOR COMPLETION ROUTINE

                                ; ***   CONTROLLER GET DRIVE STATUS VECTORS ***

007727  033752  000011  111341 V.GSTA: MOV     #GETSTA,R12     %JMP    LEV2.1  ; SEND GET STATUS COMMAND
007730  033752  000366  111445         MOV     #STARSP,R12     %JMP    R.GSTA  ; RECEIVE GET STATUS RESPONSE
007731  033757  000335  111541 V.STAT: MOV     #SDI.4&LOBYT,R17 %JMP   C.DUPL  ; GO CHECK FOR DUPLICATES
007732  034457  000017  111403 V.DCON: CLR     R17             %JMP    S.DCN1  ; SEND DISCONNECT COMMAND
007733  033753  000011  111502         MOV     #INPUT,R13      %JMP    R.RESP  ; RECEIVE DISCONNECT RESPONSE
007734  013440  000000  102074 V.ZATT: NOP                     %JMP    D.ZATT  ; CLEAT ATTN AND FINISH UP
007735  033752  000207  111341 V.GCHR: MOV     #GTCCHR,R12     %JMP    LEV2.1  ; SEND DRIVE GET COMMON CHAR
```

```
007736  033754  000010  111433          MOV     #SDI.DL,R14     %JMP    R.GCCH  ; RECEIVE DRIVE GET COMMON CHAR
007737  033752  000355  102032  V.GCR1: MOV     #UN.BUF&LOBYT,R12 %JMP  D.GSCB  ; [16K]GET SUBUNIT CHARACTERISTICS BUFFER
007740  033752  000210  101441          MOV     #GTUCHR,R12     %JMP    S.GUCH  ; SEND GET SUBUNIT CHAR COMMAND
007741  033754  000016  101443          MOV     #<SDI.LL-SDI.H1>,R14 %JMP R.GUCH ; RECEIVE SUBUNIT CHAR
007742  010555  007043  111646  V.GCR2: ADD     #SDI.UN,DBAR\N,BAR %JMP C.SUBU  ; CHECK IF ALL SUBUNIT'S CHARACTERISTICS ARE IN
007743  013440  000000  102074          NOP                             %JMP    D.ZATT  ; CLEAR ATTN AND FINISH UP

                                ; ***   DRIVE PROCESSOR DIAGNOSTIC MACHINE XFC VECTORS ***

007744  013740  007002  103367  XFCTAB: MOV     #DMREG2,BAR     %JMP    XFC01   ; [16K]XFC 01 - FORMAT TRACK ON INDEX
007745  034456  000016  103463          CLR     R16             %JMP    XFCRW   ; XFC 02 - READ N SECTORS
007746  033756  000001  113463          MOV     #1,R16          %JMP    XFCRW   ; XFC 03 - WRITE N SECTORS
007747  013740  007007  103471          MOV     #DMREG0,BAR     %JMP    XFC04   ; XFC 04 - SEND SDI COMMAND
007750  013440  000000  103477          NOP                     %JMP    XFC05   ; XFC 05 - RECEIVE SDI COMMAND
007751  013440  000000  103510          NOP                     %JMP    XFC06   ; XFC 06 - COMPARE DATA PATTERN TO BUFFER
007752  013440  000000  113530          NOP                     %JMP    XFC07   ; XFC 07 - RETURN DRIVE SIGNALS
007753  013740  007007  113541          MOV     #DMREG0,BAR     %JMP    XFC08   ; XFC 08 - ECHO DATA TO DRIVE
007754  013440  000000  113550          NOP                     %JMP    XFC09   ; XFC 09 - INITIALIZE DRIVE
007755  013440  000000  113553          NOP                     %JMP    XFC10   ; XFC 10 - WAIT FOR SECTOR/INDEX
000012                          XFCUPR  := .-XFCTAB                             ; START OF U.PROC XFC'S/END OF D.PROC'S
007756  033751  000013  113556          MOV     #11.,R11        %JMP    XFC11   ; XFC 11 - READ UNIBUS MEMORY
007757  033751  000014  103556          MOV     #12.,R11        %JMP    XFC11   ; XFC 12 - WRITE UNIBUS MEMORY
000014                          XFCPAS  := .-XFCTAB                             ; BIAS FOR U.PROC XFC'S
                                UP.XFC:                                         ; FOR EQUATE
000021                          XFCFIN  := .-UP.XFC+XFCUPR+7                    ; XFC DONE VALUE
000022                          XFCMAX  := .-UP.XFC+XFCUPR+8.                   ; MAXIMUM XFC VALUE

007760                          ASSUME XFCMAX,EQ,22
007760                          ASSUME XFCFIN,EQ,21
007760                          ASSUME XFCUPR,EQ,12

                                ; ***   NOTE - THESE HARDWARE ERROR VECTORS MUST START AT OFFOH ***
                                ; ***   D.PROCESSOR ERROR VECTORS ***

                                .ORG    ^HOFF0                                  ; *** IMPORTANT ***
007760  170451  000011  117642  CON.ER: INC\L   R11,R11          %JMP    D.STOP  ; 0 - NOT USED (BY HARDWARE)
007761  036151  000011  117642          XOR     R11,R11         %JMP    D.STOP  ; 1 - BI RD PE2
007762  036151  000011  117642          XOR     R11,R11         %JMP    D.STOP  ; 2 - BI WR PE2
007763  170451  000011  117642          INC\L   R11,R11         %JMP    D.STOP  ; 3 - NOT USED (BY HARDWARE)
007764  170451  000011  117642          INC\L   R11,R11         %JMP    D.STOP  ; 4 - NOT USED (BY HARDWARE)
007765  033711  003003  107775          MOV     (BUF),R11,BUF   %JMP    D.RAMP  ; 5 - RAM PE (ATTEMPT TO CLEAR ERROR)
007766  114544  000002  112324          BIT     #INDIAG,RLL     %JMP    D.TIMO  ; 6 - TIMEOUT / IN DIAG MODE?
007767  173751  000060  117770          MOV\L   #ER.SOP,R11     %JMP    .+1     ; 7 - CROM PE

                                ; ***   U.PROCESSOR ERROR VECTORS ***

007770  113740  004061  107774          MOV     #ER.SOP+0DDP,DCRD %JMP  D.ER    ; 8
007771  013440  000000  107642          NOP                     %JMP    D.STOP  ; 9
007772  013440  000000  107642          NOP                     %JMP    D.STOP  ; A
007773  013440  000000  107642          NOP                     %JMP    D.STOP  ; B
                                HIPAR:
007774  070151  000471  107760  D.ER:   ADD\L   R11,R11 @SCLR   %JMP    CON.ER  ; C - NOT USED (BY HARDWARE)
007775  173751  000040  107776  D.RAMP: MOV\L   #ER.SAP,R11     %JMP    .+1     ; D - NOT USED (BY HARDWARE)
007776  113740  004041  117774          MOV     #ER.SAP+0DDP,DCRD %JMP  D.ER    ; E
```



```
007777  013440  000000  107642          NOP                     %JMP    D.STOP  ; F

010000                                  .END            ;END OF FILE
```

ACL0	000007	ALGADR	034000	ALOLMT	000035	ALUTST	000166	ATTCOD	000100
ATTN	000002	AVAIL	000100	AVL.LN	000040	BA	037777	BADRH	000266
BADRL	000265	BAD.LN	000034	BAR	000007	BA16	000001	BA17	000002
BA18	000004	BA19	000010	BA20	000020	BA21	000040	BA22	000100
BA23	000200	BA24	000400	BA25	001000	BA26	002000	BA27	004000
BA28	010000	BA29	020000	BCAID	000005	BCAIS	000015	BCGRP	000002
BCICSR	000050	BDCST	000015	BDSNF	002000	BECC	010000	BECER	020000
BERDN	000400	BERMAX	000150	BFRQ	000020	BFSV	000100	BFULL	040000
BGOOD	004000	BGRUP	000001	BIBAD	000037	BIBER	000010	BICSR	000004
BICSRE	000200	BIDTYP	000000	BIEADR	000044	BIECSR	000014	BIGPRO	000360
BIGPR1	000364	BIGPR2	000370	BIGPR3	000374	BIIDES	000020	BIIMSK	000024
BIPSDE	000030	BIPSFC	000060	BIPSRC	000034	BIRDCM	000001	BIRTRY	000300
BISADR	000040	BIT00	000001	BIT01	000002	BIT02	000004	BIT03	000010
BIT04	000020	BIT05	000040	BIT06	000100	BIT07	000200	BIT08	000400
BIT09	001000	BIT10	002000	BIT11	004000	BIT12	010000	BIT13	020000
BIT14	040000	BIT15	100000	BIT16	000001	BIT17	000002	BIT18	000004
BIT19	000010	BIT20	000020	BIT21	000040	BIT22	000100	BIT23	000200
BIT24	000400	BIT25	001000	BIT26	002000	BIT27	004000	BIT28	010000
BIT29	020000	BIT30	040000	BIT31	100000	BIUCSR	000100	BIWRCM	000004
BIWSTA	000054	BLAST	100000	BLOCK	000001	BLRWR	001000	BLSTB	040000
BMAPDN	000010	BNXCOP	000004	BOARD2	001000	BRANCH	003752	BRARS	000020
BRCTS	000040	BREG	000006	BRLV4	000001	BRLV4R	000400	BRLV4S	000020
BRLV5	000002	BRLV5R	001000	BRLV5S	000040	BRLV6	000004	BRLV6R	002000
BRLV6S	000100	BRLV7	000010	BRLV7R	004000	BRLV7S	000200	BROKE	010000
BRTRY	100000	BTCNT	000236	BTROTH	004000	BUF	000003	BUFBEG	002055
BUFEND	005255	BUFLMT	000041	BUFPTR	000133	BUFP00	000133	BUFP01	000135
BUFP02	000137	BUFP03	000141	BUFP04	000143	BUFP05	000145	BUFP06	000147
BUFP07	000151	BUFP08	000153	BUFP09	000155	BUFP10	000157	BUFP11	000161
BUFP12	000163	BUFP13	000165	BUFP14	000167	BUFP15	000171	BUFP16	000173
BUFP17	000175	BUFP18	000177	BUFP19	000201	BUFP20	000203	BUFP21	000205
BUFP22	000207	BUFP23	000211	BUFP24	000213	BUFP25	000215	BUFP26	000217
BUFP27	000221	BUFP28	000223	BUFP29	000225	BUFP30	000227	BUFP31	000231
BUFP32	000233	BUFP33	000235	BUFP34	000237	BUFP35	000241	BUFP36	000243
BUFP37	000245	BUFP38	000247	BUFP39	000251	BUFP40	000253	BUFP41	000255
BUFP42	000257	BUFP43	000261	BUFP44	000263	BUFP45	000265	BUFP46	000267
BUFP47	000271	BUFP48	000273	BUFP49	000275	BUFP50	000277	BUFP51	000301
BUFP52	000303	BUFR1.	005255	BUFR2.	005672	BUF.BC	000012	BUF.BP	000002
BUF.DL	000415	BUF.EC	000401	BUF.ED	000400	BUF.GP	000011	BUF.HH	000004
BUF.HL	000003	BUF.LL	000015	BUF.NL	000000	BUF.SD	000006	BUF.ST	000001
BUF.TA	000005	BUF.UA	000007	BUF.US	000010	BUF.U1	000013	BUF.U2	000014
BUF.56	000000	B.BAD	000010	B.BTO	000004	B.CPE	000200	B.CTE	010000
B.DTYP	000416	B.ICE	000001	B.ISE	002000	B.IVE	000400	B.LED	000020
B.LOOP	000100	B.MPE	004000	B.MTCE	020000	B.NABO	000001	B.NEX	000002
B.NMR	040000	B.NRTY	000200	B.ODD	000040	B.RDS	000040	B.RTO	000020
B.SPE	000100	B.STO	000010	B.TDF	001000	B2TST	000161	CCLASS	000400
CCSRWO	000032	CCSRW1	000033	CDONE	000016	CDUPLA	001543	CDUPLB	001550
CDUPLC	001562	CDUPLD	001563	CDUPLE	001566	CFLAGS	000320	CF.ATN	000200
CF.MSC	000100	CF.THS	000020	CHGFLG	000202	CHGMOD	000201	CKSTO	003713
CLIMIT	000142	CLK	000464	CLKP	000463	CLK00	000472	CLK01	000505
CLK02	000504	CLK1	000462	CLK10	000461	CLK16	000460	CLK32	000456
CLK6	000457	CLK64	000455	CLRERR	000165	CLRTST	000051	CMDCOF	000030
CMDLEN	000026	CMDLIM	000024	CMDO	000004	CMDPOF	000031	CMDPTR	000027
CMODEL	000022	CMPBUF	000242	CNTFLG	000170	CNT.LN	000030	CNVTV1	000040
CNVTV2	000042	CNVTV3	000044	CNVTV4	000045	CN.ERR	004000	CODVER	000023
COMPLT	000176	CONT	000001	CONTCD	152000	CONT.F	001333	CON.ER	007760
CON.ST	000170	COPY4	000000	CPE	000012	CR	000004	CRDY	000010
CRI	000004	CRY	000022	CSEEKA	001675	CSERR	000010	CSR	000031
CTMOUT	000170	CYCLE	000552	CYLSTR	170000	C$TYPE	000002	C.AONL	001624
C.ATTN	001630	C.DCON	001665	C.DUPL	001541	C.DUPX	001627	C.IMFF	001613
C.LOGA	001636	C.LOGS	001633	C.SEEK	001673	C.SUBC	001660	C.SUBU	001646
DATEND	033317	DATT	000400	DBAR	000015	DCERR	000001	DCLASS	001000

DCLK	000030	DCMASK	000170	DCN.ST	000001	DCN.TT	000200	DCRD	000004
DCRS	000001	DCRTST	000251	DDC	002000	DDD	004000	DER	000011
DERR	000004	DEVCL	000377	DFAIL	002000	DHANG	000046	DI	000400
DIAG	000400	DINITA	001024	DINITB	001027	DINITC	001030	DISCON	000204
DISM	000062	DISN	000063	DL11AD	177560	DL1144	176510	DMBEG	020000
DMBPC	000002	DMDBG	000000	DMDT	004020	DMNTRY	003626	DMODE	040000
DMODT	000012	DMOVH	000011	DMOVL	000010	DMPC	003632	DMREGO	000007
DMREG1	000001	DMREG2	000002	DMREG3	000003	DMREG4	000004	DMREG5	000005
DMREG6	000006	DMREG7	000000	DMSTR	000013	DMTEMP	000166	DM.BEG	001352
DOFFDA	002053	DOFFDB	002061	DPF	000026	DPROC	000006	DRAMPE	000304
DRAVL	010000	DRDUP	002000	DRDY	000001	DRINIT	004000	DROMPE	000266
DRPLCA	001067	DRPLCB	001107	DRPLCC	001112	DRPLCD	001127	DRPLCX	001105
DRPLC1	001141	DRVCLR	000005	DRVOL	004000	DRV.AT	002000	DRV.AV	004000
DRV.C1	004000	DRV.C2	002000	DRV.C3	001000	DRV.C4	000400	DRV.DB	001000
DRV.DD	004000	DRV.DE	000200	DRV.DF	000020	DRV.DR	000040	DRV.EL	000010
DRV.FO	002000	DRV.OA	000200	DRV.PE	000040	DRV.PS	000002	DRV.RE	000100
DRV.RR	000100	DRV.RU	000001	DRV.SN	170000	DRV.SR	000020	DRV.SU	170000
DRV.S1	010000	DRV.S2	020000	DRV.S3	040000	DRV.S4	100000	DRV.S7	000400
DRV.UM	176000	DRV.U1	010000	DRV.U2	020000	DRV.U3	040000	DRV.U4	100000
DRV.WE	000010	DRV.W1	010000	DRV.W2	020000	DRV.W3	040000	DRV.W4	100000
DSDIEA	001744	DSDIEB	001763	DSDIEC	001753	DSER	000034	DSK.LN	000054
DSLEDS	000151	DSR	000035	DSTSH	000007	DSTSL	000006	DTEMP1	000206
DTIMEA	002113	DTMP	000012	DTSTBL	007404	DUPVC	001000	DUP.LN	000014
DXFC	010000	D.ALBF	003006	D.ARTC	003200	D.ATTN	001053	D.AVAL	001054
D.BFCA	002476	D.BFCK	002475	D.CLCS	001270	D.CLRS	002011	D.CLUR	002306
D.CPUL	002507	D.DBNO	003254	D.DIAG	000053	D.DIVD	003257	D.DNSK	002016
D.DONE	002022	D.DONZ	002015	D.DTER	002341	D.DVDA	003261	D.DVDB	003262
D.DVDC	003264	D.DVDD	003265	D.DVDE	003267	D.DVDF	003271	D.DVDG	003272
D.ECKA	001721	D.ECKB	001723	D.ECKC	001730	D.ECKD	001735	D.ECKR	001704
D.ECKS	001714	D.ECRA	001711	D.ECRB	001712	D.EERR	002343	D.END	000131
D.ER	007774	D.ERCK	001715	D.ERRD	001002	D.ERRO	001773	D.ERR1	000767
D.EROA	002006	D.ER1A	000776	D.E.C.	000003	D.GECC	003002	D.GGRP	002544
D.GMST	003572	D.GPRE	002502	D.GRPC	002537	D.GSCB	002032	D.GUNF	002023
D.HCEA	002401	D.HCEC	002415	D.HCED	002422	D.HCER	002352	D.HCEO	002361
D.IDLA	000700	D.IDLB	000707	D.IDLC	000727	D.IDLD	000732	D.IDLE	000701
D.IDLF	000736	D.IDLG	000742	D.IDLH	000743	D.IDLI	000745	D.IDLJ	000747
D.IDLM	000752	D.IDLN	000755	D.INIT	001022	D.INSV	002037	D.INSO	002005
D.INS1	002036	D.INS2	001672	D.IODA	002251	D.IODB	002252	D.IODC	002265
D.IODD	002272	D.IODN	002231	D.IOEA	002455	D.IOER	002424	D.IOEX	002305
D.IOPR	002467	D.IOPW	002465	D.LDHD	003053	D.LOGF	002043	D.LRER	002342
D.NCLK	001766	D.NOBF	002310	D.OFFD	002045	D.OFFL	002064	D.OLRD	002071
D.ONLN	001055	D.OVER	002323	D.PCMB	001045	D.PCMD	001034	D.RAMP	007775
D.RAMX	000704	D.RBLH	002460	D.RCLR	001271	D.RCTE	002340	D.READ	002600
D.RECC	002744	D.REDA	002602	D.REDB	002613	D.REDC	002620	D.REDE	002633
D.REDF	002673	D.REDG	002677	D.REDH	002700	D.REDI	002701	D.REDJ	002764
D.REDL	002773	D.REDM	002775	D.REDN	002776	D.REDX	002606	D.RED1	002632
D.RLF1	002653	D.RLF2	002671	D.RNBF	002670	D.RPLC	001061	D.RRDE	002335
D.RRDY	000525	D.RRWE	002333	D.RSAV	003365	D.RSTR	000002	D.RTDE	002327
D.RVCA	001151	D.RVCT	001142	D.RVC2	003273	D.RVC4	003323	D.RVC6	003323
D.RVC8	003342	D.RVDX	003321	D.RV2A	003317	D.RV4A	003335	D.RV4B	003337
D.RV8A	003353	D.RWRA	002332	D.RWRE	002331	D.SDIE	001742	D.SDIF	001770
D.SDIG	001771	D.SEEK	001163	D.SETA	002303	D.SETB	003005	D.SETS	002302
D.SGRP	002535	D.SLBA	003234	D.SLBN	003233	D.SMOV	002105	D.SMVX	002104
D.SRTC	003055	D.SSET	002013	D.STOP	007642	D.STPA	007640	D.SUCH	001164
D.TIME	002112	D.TIML	002121	D.TIMO	002324	D.TIMR	002110	D.TKEA	002337
D.TKER	002336	D.TOER	002326	D.TOPO	001165	D.VECA	001167	D.VECB	001206
D.VECP	001166	D.VECT	001170	D.WAIT	002073	D.WDAT	002570	D.WDTB	002572
D.WDTC	002574	D.WDTD	002576	D.WHDA	002552	D.WHDB	002554	D.WHDC	002555
D.WHDR	002550	D.WRIT	002123	D.WRTB	002133	D.WRTC	002140	D.WRTE	002146
D.WRTF	002176	D.WRTG	002201	D.WRTI	002223	D.WRTJ	002227	D.WRTX	002130
D.WSAF	002520	D.WSGA	002515	D.WSIG	002513	D.ZATT	002074	ECC	000007
ECCMSK	176000	ECCRD	000440	ECC9.1	000035	ECC9.2	000046	ECC9.3	000057

LSCS FORM=QUAD

ECHOCD	164000	ECOUNT	000134	ECSUMH	022000	ECSUML	000305	EDSEED	000105
EF.BBR	100000	EF.BBU	040000	EF.LOG	020000	ELEV	100000	ELOC	000070
ELPM	000046	ELPN	000035	ELPO	000070	ELPP	000057	ENCACH	033737
ENDCD	131000	END.F	001337	EQ	000021	ERECOV	000006	ERR	100000
ERRBIT	100000	ERRB1A	000031	ERRB1E	000032	ERRB2	000040	ERRB2E	000035
ERRB2H	000036	ERRB2I	000037	ERREG	000017	ERRINI	040000	ERRINP	100000
ERRIP	040000	ERRRTC	000004	ERRSET	000042	ERRVEC	000377	ERR00	104000
ERR01	000040	ERR02	000100	ERR03	000140	ERR04	000200	ERR05	000240
ERR06	000300	ERR07	000340	ER.BCA	000147	ER.BP1	000003	ER.DMX	000014
ER.HTO	000011	ER.INT	000010	ER.IWR	000017	ER.MER	000144	ER.MRR	000026
ER.MST	000013	ER.NIM	000012	ER.PRD	000001	ER.PWR	000002	ER.RAP	000004
ER.ROP	000005	ER.RP2	000003	ER.RRD	000006	ER.RTO	000150	ER.RWR	000007
ER.SAP	020000	ER.SOP	030000	ER.SRP	010000	ER.STP	000146	ER.TMO	000015
ER.VCI	000016	ER.WP2	000003	ETST1	000013	EVAL	000046	FAILUR	000034
FAIRCT	077400	FCT.MD	000000	FCT.V1	000002	FCT.V2	000003	FDCLK	000020
FDIAG	100000	FETCH	003631	FETCH2	003632	FETCH3	003634	FLAG	040000
FLOAT	000010	FM.BAD	000001	FM.CNT	000000	FM.DSK	000002	FM.SDI	000003
FORCE	000001	FORICD	025400	FORSCD	046400	FPE	000200	FRERR	000004
FRMCD	177400	FSEEK	000004	FSTST	000114	FTEST	000010	GENSYN	000533
GETCRY	003630	GETECC	000537	GETECO	000535	GETRB	003702	GETRBX	003703
GETSTA	000011	GO	000001	GORD	000001	GOWR	000012	GPRO	010000
GPR1	020000	GPR2	040000	GPR3	100000	GSEL	001000	GTCCHR	000207
GTCRSP	000170	GTE	000022	GTUCHR	000210	GTURSP	000167	HADD	000017
HANG	000003	HDCOD	170000	HDLMT	000136	HDRAD1	003137	HDRANA	003103
HDRANB	003121	HDRANC	003125	HDRAND	003131	HDRANE	003150	HDRANL	003064
HDRCCH	003245	HDRCCL	003243	HDRCHK	003247	HDRCKE	003253	HDRCKH	003246
HDRCKL	003244	HDRCMD	003023	HDRCME	003024	HDRCMF	003050	HDRCMG	003052
HDRCMP	003014	HDRNOM	003071	HDRTMO	001000	HDSLB1	003207	HDSUBA	003204
HD.TO	000053	HD.T1	000053	HD.T2	000053	HD.T8	000100	HEADER	000010
HES	100000	HIBYT	177400	HINIB	000360	HIPAR	007774	HI.OFF	000374
HI1	003031	HI2	003035	HI3	003041	HI4	003045	HOSTF	000377
HOSTMO	000200	HSTQUE	000204	H11COD	110000	H11XOR	120000	IDENT	000011
IDX	000040	IE	000200	IMMED	003645	INCRTN	000250	INDIAG	001000
INDRCT	003750	INI	000020	INITM1	003625	INIT1	003624	INPLEN	000007
INPUT	000011	INSEEK	000012	INSTR	000265	INTI	000200	INTR	000010
INTVEC	000164	INVAL	000014	IOACC	020000	IOCLK	010000	IOCNT	002000
IOMSK	000060	IORTY	036000	IORWR	020000	IOSEK	004000	IPINTR	000016
IPREG	000362	IRCI	000002	IUAR	000006	JCOND	004011	JEQJPL	004013
JMPTST	000005	JNEJMI	004016	JSBXFC	003765	JSIGN	004015	JUMP	000015
LAB	000000	LADD	000007	LATE	000036	LBNMSK	000377	LBNSTR	007400
LBWR	000016	LCOM	000006	LDRT1	003610	LEDS	170000	LED1	010000
LED2	020000	LED4	040000	LED8	100000	LESS	000002	LEVORA	001250
LEVORB	001253	LEVORC	001257	LEVORD	001244	LEVORE	001261	LEVORF	001263
LEVORG	001266	LEVORX	001260	LEVOR2	001251	LEVOWA	001213	LEVOWE	001225
LEVOWR	001212	LEVOWS	001226	LEVOWW	001207	LEVOWX	001210	LEV1RA	001303
LEV1RB	001304	LEV1RC	001317	LEV1RD	001273	LEV1RE	001321	LEV1RF	001330
LEV2TA	001352	LEV2.E	001345	LEV2.T	001347	LEV2.1	001341	LEV2.2	001343
LFAIL	001000	LF.CON	040000	LF.SNR	000400	LF.SUC	100000	LGADR	036000
LGCKSV	000276	LM	000064	LN	000240	LNERR	000020	LNPAG	000002
LOBYT	000377	LOGCOD	000100	LOGEND	000342	LOGLEN	000154	LOGPKT	000304
LOG.LN	000034	LONIB	000017	LOPAR	007403	LO.OFF	000003	LO1	003027
LO1A	003030	LO2	003033	LO3	003037	LO4	003043	LSB	000004
LT400	000000	LV1SV1	000212	LV1SV2	000214	LV2CNT	001400	LW	040000
L.BADO	000016	L.BAD1	000017	L.CHVR	000014	L.CNTO	000010	L.CNT1	000011
L.CNT2	000012	L.CNT3	000013	L.CRFO	000002	L.CRF1	000003	L.CSVR	000014
L.EVNT	000007	L.FLGS	000006	L.FMT	000006	L.HDRO	000026	L.HDR1	000027
L.MLUN	000015	L.RTRY	000023	L.SDIO	000030	L.SDI1	000031	L.SDI2	000032
L.SDI3	000033	L.SDI4	000034	L.SDI5	000035	L.SEQ	000005	L.UHVR	000022
L.UNIT	000004	L.UNTO	000016	L.UNT1	000017	L.UNT2	000020	L.UNT3	000021
L.USVR	000022	L.VSEO	000024	L.VSE1	000025	M	000061	MAPENB	100000
MAPFLG	000246	MAPMSC	000077	MAPMSH	077700	MAPMSK	000003	MAPSAV	000244
MAPVAL	100000	MAP.CH	000101	MAP.MO	000103	MAP.M1	000104	MAP.ND	000116

MAP.NX	000102	MAP.OF	000107	MAP.RD	000105	MAP.ST	000115	MAP.S1	000110
MAP.UR	000106	MAP.VO	000111	MAP.V1	000112	MAP.V2	000113	MAP.V3	000114
MAXSST	000037	MAXTIM	177777	MCP.IO	000020	MCP.LN	000030	MCP.RD	000022
MD.ALL	000002	MD.CMP	040000	MD.CSE	020000	MD.ERR	010000	MD.EXC	000040
MD.EXP	100000	MD.FEU	000001	MD.FKC	000001	MD.IMF	000002	MD.NOV	020000
MD.NXU	000001	MD.PRI	000001	MD.RIP	000001	MD.SAV	000004	MD.SCH	004000
MD.SCL	002000	MD.SEC	001000	MD.SEQ	000020	MD.SER	000400	MD.SHD	000020
MD.SPD	000001	MD.SSH	000200	MD.SWP	000004	MD.VOL	000002	MD.WBN	000100
MD.WBV	000040	MD512	126736	MEMDST	003742	MEMSZ	000042	MERR	000014
MINSST	000003	MINTIM	141520	MINUTE	000074	MODEO4	003706	MODE4	003711
MODE6	003676	MODNUM	000440	MP	000100	MRQUE	000146	MSB	000023
MSCPLN	000264	MWQUE	000150	NACLO	000027	NBCB	000200	NBIBAD	000017
NBUFR	000051	NCDONE	000036	NCPE	000032	NCRY	000002	NCSR	000011
NDCLK	000010	NDPF	000006	NDSER	000014	NDSR	000015	NEG	000023
NENDCD	046400	NEQ	000001	NETST1	000033	NFTEST	000030	NLATE	000016
NLSB	000024	NMERR	000034	NMSB	000003	NNEG	000003	NODEID	000017
NOPREG	000000	NOVER	000016	NPKTS	000024	NPOLL	000035	NPRDY	000010
NRPE	000033	NRPOK	000007	NRRDY	000012	NSCAN	000035	NSCB	000010
NSDI	000004	NSECTR	000032	NSEEKS	000156	NSTOP	000031	NTEST	000025
NUMFLG	000001	NUPF	000006	NWRC	000017	NZRO	000001	OBCLR	003736
OBSET	003731	OBTST	003726	OCLK	000511	OCLKRS	000506	OCLKOO	000513
OCLK1	000510	OCLK48	000507	OCOMP	003725	ODDP	000400	ODTAF	000003
ODTCA	000005	ODTCY	000004	OFFMSK	000777	OFFSET	000020	OFFTRK	000002
OJCC	003764	OLOAD	003723	ONLINE	000213	ONLREC	100000	ONLY	000536
OPCOD	000377	OPCODE	000144	OPDCD	003716	OPDCD1	003717	OPDCD2	003720
OPM	000200	OPNFLG	000002	OPXXO	003722	OPXX1	003727	OPXO1	003730
OPX10	003724	OPX11	003735	OP.ABO	000001	OP.ACC	000020	OP.ACP	000102
OP.ATT	000017	OP.AVA	000100	OP.AVL	000010	OP.CCD	000021	OP.CMP	000040
OP.DAP	000013	OP.DUP	000101	OP.END	000200	OP.ERS	000022	OP.FLU	000023
OP.GCS	000002	OP.GST	000016	OP.GUS	000003	OP.MRD	000030	OP.MWR	000031
OP.ONL	000011	OP.RD	000041	OP.RPL	000024	OP.SCC	000004	OP.SUC	000012
OP.WR	000042	OSUBT	003737	OTADD	003732	OVER	000036	OW	140000
OWN	100000	OXFC	003772	OXFCO	004003	OXFC17	003776	P	000002
PABRT	000001	PACTV	040000	PAGESZ	001000	PAR	000002	PARTST	000240
PCREL	003704	PKIP	000001	PKTBUF	000342	PKTEND	001352	PKT001	000342
PKT002	000374	PKT003	000426	PKT004	000460	PKT005	000512	PKT006	000544
PKT007	000576	PKT008	000630	PKT009	000662	PKT010	000714	PKT011	000746
PKT012	001000	PKT013	001032	PKT014	001064	PKT015	001116	PKT016	001150
PKT017	001202	PKT018	001234	PKT019	001266	PKT020	001320	PLOCK	100000
POLL	000015	PONER	000002	PRDY	000030	PSTACK	000052	PSTAT	100000
PTELEN	000002	P.BCNO	000010	P.BCN1	000011	P.BUFH	000013	P.BUFL	000012
P.BUFO	000012	P.BUF1	000013	P.BUF2	000014	P.BUF3	000015	P.BUF4	000016
P.BUF5	000017	P.CMSO	000012	P.CMS1	000013	P.CNTF	000011	P.CNTO	000014
P.CNT1	000015	P.CNT2	000016	P.CNT3	000017	P.CONS	000004	P.CRFO	000002
P.CRF1	000003	P.CSVR	000013	P.CTMO	000012	P.CYL	000026	P.CYLS	000026
P.ELGO	000020	P.ELG1	000021	P.FBBO	000020	P.FBB1	000021	P.FLGS	000006
P.GRP	000025	P.HSTO	000012	P.HST1	000013	P.HTMO	000012	P.LBNO	000020
P.LBN1	000021	P.LCKA	003566	P.LINK	000000	P.LOCK	003567	P.MED1	000020
P.MED2	000021	P.MLUN	000010	P.MOD	000007	P.OPCD	000006	P.RBNS	000031
P.RBNO	000010	P.RBN1	000011	P.RCTC	000031	P.RCTS	000030	P.RFNO	000010
P.RFN1	000011	P.RGIO	000020	P.RGI1	000021	P.RGOO	000022	P.RGO1	000023
P.RSO3	000005	P.RSO6	000010	P.RSO8	000012	P.RS16	000022	P.RS17	000023
P.RS19	000025	P.SHST	000023	P.SHUN	000022	P.STS	000007	P.TRCK	000024
P.UNFL	000011	P.UNIT	000004	P.UNSO	000024	P.UNS1	000025	P.UNTO	000014
P.UNT1	000015	P.UNT2	000016	P.UNT3	000017	P.USVR	000027	P.VCID	000001
P.VRSN	000010	P.VSEO	000026	P.VSE1	000027	Q	000000	QB	001000
QDADM	000001	QREVB	000001	QUESAV	000152	QW	100000	Q.STAT	140000
RAMPE	000100	RAMPET	000315	RAMTST	000322	RAMZAP	000311	RBCAI	000013
RBNCLA	003222	RBNCLB	003226	RBNCLC	003211	RBNCOD	060000	RBNMSK	000077
RBNPRI	003210	RBNPRM	050000	RBNSTR	007400	RBNXOR	060000	RBN2ND	030000
RBPA	000017	RCC	000014	RCI	000003	RCLR	000013	RCMD	000012
RCSRWO	000024	RCSRW1	000025	RCTBLK	000224	RCTOFF	000222	RCT.EN	100000

RCT.MT	000000	RCT.PR	020000	RCT.SE	030000	RCY	000015	RD	000004
RDCLK	000500	RDCLKO	000476	RDFST	000004	RDNXT	000014	RDPF	000010
READ	000001	READCD	013400	RECALB	000216	RECC	000006	RECCT	000007
RELES	000136	REPSTA	000280	RETCNT	007400	RETURN	003760	REVEND	000133
REVSTR	000123	RGDATH	000272	RGDATL	000271	RIB	000013	RLL	000004
RNGBSH	000162	RNGBSL	000160	ROL	003657	ROR	003662	ROTATE	003652
ROTR1	003675	ROTR3	003674	RPC	000012	RPE	000013	RPOK	000027
RRDY	000032	RRM	000005	RRSGEN	000001	RSE	000004	RSPCOF	000022
RSPLEN	000020	RSPO	000002	RSPPOF	000023	RSPPTR	000021	RTCS	000005
RTCSO5	000040	RTCSO6	000100	RTCS13	000040	RTCS14	000100	RTDS	000002
RTDSER	001020	RTDSO6	000100	RTOEVE	000010	RT1	000016	RUNN	000014
RUPF	000010	RVACTV	000020	RVCBNH	000131	RVCBNL	000130	RVCBUF	000226
RVCEBF	000230	RVCFLG	000254	RVCIND	000216	RVCRTY	000220	RVCSAV	000123
RVCSDI	000232	RVCT	100000	RVCTA	000132	RVCVEC	000234	RVECDP	000252
RVECUP	000250	RVWRIT	000001	RWM	000015	RWRDY	100000	R.GCCH	001433
R.GSTA	001445	R.GSTB	001452	R.GUCH	001443	R.GXCH	001436	R.LRSP	001476
R.RESP	001502	R.RSPA	001503	R.SEEK	001536	R.TOPO	001445	R0	000000
R1	000001	R10	000010	R11	000011	R12	000012	R13	000013
R14	000014	R15	000015	R16	000016	R17	000017	R2	000002
R3	000003	R4	000004	R5	000005	R6	000006	R7	000007
SAREG	000364	SAVADR	000274	SAVBUF	000270	SAVCNT	000272	SAVEDC	000266
SAVR7	000172	SAVUAR	000262	SCAN	000015	SCLR	000003	SCMD	000002
SC.AOL	000400	SC.BAD	000300	SC.CNT	000140	SC.DCL	000240	SC.DDE	000340
SC.DIS	000400	SC.DSY	000140	SC.DUP	000200	SC.ECC	000340	SC.EC1	000400
SC.EC2	000440	SC.EC3	000500	SC.EC4	000540	SC.EC5	000600	SC.EC6	000640
SC.EC7	000700	SC.EC8	000740	SC.EDC	000100	SC.FER	000000	SC.HDR	000100
SC.IMR	000240	SC.INV	000100	SC.IOP	000100	SC.LVO	000400	SC.NML	000000
SC.NOM	000140	SC.NVL	000040	SC.N12	000240	SC.ODB	000100	SC.ODT	000040
SC.OVR	000040	SC.PAR	000200	SC.POS	000140	SC.RRD	000300	SC.RWR	000200
SC.SCN	000100	SC.SDI	000040	SC.STO	000040	SC.UNK	000000	SC.WPH	020000
SC.WPS	010000	SD	000006	SDCLK	000501	SDIBEG	001355	SDIB.L	000120
SDICLK	000345	SDICL1	000350	SDICL2	000351	SDIRTY	000003	SDIS	000017
SDITMO	000017	SDITO	004400	SDITST	000354	SDI.AT	000048	SDI.BH	000030
SDI.BL	000027	SDI.CH	000012	SDI.CL	000011	SDI.CP	000002	SDI.CW	000056
SDI.DB	000007	SDI.DL	000010	SDI.DP	000070	SDI.EC	000035	SDI.ER	000035
SDI.ES	000026	SDI.EO	000020	SDI.E1	000017	SDI.FC	000073	SDI.GC	000065
SDI.GO	000074	SDI.GP	000013	SDI.H1	000063	SDI.H2	000064	SDI.I1	000037
SDI.I2	000040	SDI.I3	000041	SDI.LL	000101	SDI.LN	000070	SDI.LT	000034
SDI.L1	000075	SDI.L2	000076	SDI.M1	000071	SDI.M2	000072	SDI.OE	000024
SDI.OM	000025	SDI.OP	000054	SDI.PH	000100	SDI.PL	000055	SDI.PQ	000022
SDI.RC	000034	SDI.RH	000032	SDI.RL	000031	SDI.RO	000023	SDI.RS	000036
SDI.RT	000067	SDI.RV	000077	SDI.SD	000057	SDI.SL	000001	SDI.SS	000014
SDI.ST	000000	SDI.SV	000021	SDI.SW	000003	SDI.S1	000044	SDI.S2	000045
SDI.S4	000050	SDI.S5	000051	SDI.S6	000052	SDI.S7	000053	SDI.TG	000066
SDI.TI	000042	SDI.TM	000004	SDI.TO	000033	SDI.UB	000006	SDI.UE	000043
SDI.UF	000060	SDI.UG	000005	SDI.UN	000043	SDI.V1	000061	SDI.V2	000062
SDI.XH	000016	SDI.XL	000015	SDI.XR	000033	SDI.1	001355	SDI.1T	000010
SDI.12	000074	SDI.2	001475	SDI.2T	000047	SDI.3	001615	SDI.4	001735
SDPF	000000	SDRD	000430	SDRRDY	000522	SDTIMO	000200	SDWRT	000374
SD.RS	000516	SD.RST	000514	SEC	000020	SECC	000016	SECCT	000017
SECSZ	000400	SECTR	000012	SEEK	000002	SEEKGS	001526	SEEKL	000006
SEKPRV	000051	SEKSAV	000052	SEQERR	000003	SEQTST	000022	SEQUEN	000010
SEQ01	000030	SERIAL	000040	SES	040000	SETCRY	003734	SEX2SK	000036
SF	000400	SFERR	000002	SFT20	000214	SGRPCD	107000	SIMTST	000000
SLAT	020000	SM	000040	SNGLOP	003647	SRM	000015	SRSGEN	000011
SSE	000014	SST	002000	STARSP	000366	START	000432	STCACH	033317
STDALN	000400	STEP	000011	STEP1	004000	STEP2	010000	STEP3	020000
STEP4	040000	STEST	000030	STOP	000011	STOPCM	140000	STOPEN	020000
STORB	003747	STORCC	003673	STORE	003744	STORT1	003621	STRTCD	070400
STRT.F	001332	STS	004000	ST.ABO	000002	ST.AVL	000004	ST.CMD	000001
ST.CMP	000007	ST.CNT	000012	ST.DAT	000010	ST.DRV	000013	ST.HST	000011
ST.MFE	000005	ST.MSK	000037	ST.OFL	000003	ST.SHF	000007	ST.SUB	000040

ST.SUC	000000	ST.WPR	000006	SUPF	000000	SUSP	000040	SWAB	003666
SWM	000005	SY	000112	SYNC	000202	SYNCH	023000	SYNCL	000274
SYO	000112	SY8	000122	S.ADRH	000023	S.ADRL	000022	S.CFL1	001367
S.CLRB	003623	S.CL11	002015	S.CMOD	001360	S.CYLH	000025	S.CYLL	000024
S.DCLR	001401	S.DCNE	001413	S.DCN1	001403	S.DCN2	001414	S.DCN3	001411
S.EREC	001426	S.GRUP	000026	S.GUCH	001441	S.INIT	001465	S.LBNH	000017
S.LBNL	000016	S.LDCC	003606	S.LDCY	003607	S.LDDB	003607	S.LDPC	003604
S.LDR2	003602	S.LD11	003603	S.LD12	003604	S.LD13	003605	S.LD14	003606
S.LD15	003607	S.LD16	003610	S.LD17	003611	S.LL12	003612	S.LRSP	001506
S.ML12	003613	S.ONLN	001467	S.OPFL	000031	S.RCAL	001471	S.ROTL	003574
S.RR17	003577	S.RUNN	001504	S.SD11	003600	S.SECI	000031	S.SECS	000030
S.SEEK	001510	S.SND3	001375	S.SRM	000531	S.STCC	003617	S.STDB	003620
S.STPC	003615	S.ST11	003614	S.ST12	003615	S.ST13	003616	S.ST14	003617
S.ST15	003620	S.ST16	003621	S.ST17	003622	S.SWAB	003575	S.TRAK	000027
TEMP	000256	TEMP1	000046	TEMP2	000047	TEMP3	000050	TEST	000005
TESTBL	000004	THEN	000451	TMOUT	000276	TMR.BS	000176	TMR.MC	000174
TOPOL	000220	TOPRSP	000157	TPO10	000411	TPO11	000417	TPO12	000442
TPO13	000451	TPO14	000432	TPO15	000536	TPO16	000432	TPO17	000503
TSBRPE	000264	TSCRPE	000013	TSTCNT	000000	TSTHNG	000014	TSTRTN	000524
TSTXL	000142	T3	000054	T4	000060	T5	000064	T6	000067
T7	000075	T8	000101	UBAR	000005	UBURST	000140	UCRD	000004
UCRS	000014	UCSREN	000400	UDA1	000001	UDD	000005	UDDI	000010
UDS	000015	UER	000007	UF.CMR	000001	UF.CMW	000002	UF.MSK	000003
UF.RMV	000200	UF.RPL	100000	UF.WPH	020000	UF.WPS	010000	UF.576	000004
UMP	000020	UMPN	000000	UNC.RM	000200	UNC.SS	100000	UNERR	000040
UNITF	177400	UNLOCK	003570	UNSUCC	000175	UN.BUF	001355	UPF	000026
UPROC	000026	UP.XFC	007760	UQ.CNT	000000	URETRY	000040	UTEMP1	000210
UTMP	000010	UWMCI	000006	U8K	000100	VAXINT	000400	VAXO	000006
VAX.PG	000400	VC.CMD	000000	VC.DMM	177760	VC.LOG	000020	VC.RSP	000000
VECT	000010	VECTAB	007643	V.ATTN	007655	V.ATT2	007712	V.AVAL	007721
V.CMD1	007675	V.DCLR	007657	V.DCON	007732	V.DRVC	007643	V.ERRC	007661
V.ERRO	007655	V.ERR1	007705	V.ERSK	007664	V.GCHR	007735	V.GCR1	007737
V.GCR2	007742	V.GSTA	007727	V.INIT	007654	V.NULL	007710	V.OLRD	007711
V.ONAV	007724	V.ONLN	007711	V.RVC3	007644	V.RVC5	007645	V.RVC8	007646
V.SEEK	007702	V.SKON	007665	V.STAT	007731	V.SUCH	007714	V.TOPO	007651
V.ZATT	007734	WAIT	000156	WCI	000005	WD	000000	WMCI	000007
WRAP	000000	WRC	000037	WRFST	000005	WRITCD	122400	WRITE	000004
WRNXT	000015	WRRDXR	131000	WRRDX1	175377	WRSND	000003	WRT	000002
XBNCOD	120000	XBNSTR	170000	XCMP	000200	XFCFIN	000021	XFCMAX	000022
XFCPAS	000014	XFCRET	004006	XFCRTA	004010	XFCRW	003463	XFCSST	003454
XFCTAB	007744	XFCUPR	000012	XFCO1	003367	XFCO2	003463	XFCO3	003463
XFCO4	003471	XFCO5	003477	XFCO6	003510	XFCO6A	003516	XFCO7	003530
XFCO8	003541	XFCO8A	003544	XFCO8B	003547	XFCO9	003550	XFC1A	003407
XFC1ER	003462	XFC1O	003553	XFC11	003556	XFC12	003566	XFC5A	003500
XFC5B	003506	XF.REP	003405	XF04A	003474	XF11A	003561	XF11B	003562
XF11C	003564	XF11D	003565	XRWRER	003461	XSSTAA	001016	X.SSDI	001004
X.SSDS	001012	X.SSDT	001006	X.SSDX	001007	X.SSTA	001015	ZFLG	000010
ZRO	000021	$DPASS	000131	$DRPE	000271	$ECC	000434	$END	010000
$SDI1	000346	$SDI2	000347	$SDI3	000352	$SDI4	000372	$SDR1	000520
$SDR2	000521	$SDR3	000523	$SD1	000402	$SD2	000403	$SD3	000404
$SD4	000413	$SD5	000423	$$BDA	000002	$$KDA	000001	$$KDB	000002
$$QDA	000001	$$UDA	000000						

LSCS FORM=QUAD

CROSS REFERENCE TABLE (CREF V01-08)

```
$$BDA     1-0      8-0     12-0     13-0     14-0     14-0     15-0     15-0     18-0     22-0     30-0     38-0     45-0     46-0
         51-0     51-0     51-0     51-0     64-0     65-0     67-0     71-0     74-0     74-0     74-0     76-0     76-3     79-31
         80-44    83-73    83-103   83-114   91-251   91-251   91-253   119-541  176-2544 224-7400 225-7416 226-7416 228-7760
$$KDA     1-0      1-0
$$KDB     1-0      1-0
$$QDA     1-0      1-0      8-0     12-0     13-0     15-0     15-0     18-0     30-0     45-0     46-0     63-0     65-0     65-0
         65-0     65-0     71-0     76-4     83-104   83-113   85-142   121-701  185-3055 224-7404
$$UDA     1-0      8-0     12-0     13-0     15-0     15-0     18-0     20-0     21-0     22-0     22-0     25-0     27-0     29-0
         45-0     46-0     59-0     67-0     70-0     74-0     74-0     75-0     83-113   93-266   94-302   99-353   100-362  101-373
        111-467  116-521  117-527  202-3531
$DPASS   84-131
$DRPE    93-271
$ECC    108-434
$END    229-8000
$SD1    104-402
$SD2    104-403
$SD3    104-404
$SD4    106-413
$SD5    106-423
$SDI1    99-346
$SDI2    99-347
$SDI3    99-352
$SDI4   101-372
$SDR1   116-520
$SDR2   116-521
$SDR3   116-523
ACLO     12-0     13-0     13-0    121-701  185-3055
ALGADR   65-0     65-0     65-0
ALOLMT   48-0     48-0
ALUTST   82-54    88-166
ATTCOD   45-0
ATTN     25-0     99-353  100-362  121-732  122-740  122-742  122-743  122-753
AVAIL    25-0     99-353  101-371  121-722  121-732  121-733  122-740  122-743  122-750  122-752  153-1606 157-1745 161-2100
AVL.LN   52-0
B.BAD    30-0
B.BTO    42-0
B.CPE    42-0
B.CTE    42-0
B.DTYP   42-0
B.ICE    42-0
B.ISE    42-0
B.IVE    42-0
B.LED    30-0
B.LOOP   30-0
B.MPE    42-0
B.MTCE   42-0
B.NABO   30-0
B.NEX    42-0
B.NMR    42-0
B.NRTY   30-0
B.ODD    30-0
B.RDS    42-0
B.RTO    42-0
B.SPE    42-0
```


CROSS REFERENCE TABLE (CREF V01-08)

```
B.STO    42-0
B.TDF    42-0
B2TST    83-63    87-161
BA       21-0     30-0     43-0
BA16     21-0     30-0     43-0     43-0
BA17     21-0     30-0     43-0     43-0
BA18     30-0     42-0     43-0
BA19     30-0     42-0     43-0
BA20     30-0     42-0     43-0
BA21     30-0     42-0     43-0
BA22     42-0     43-0
BA23     42-0     43-0
BA24     42-0     43-0
BA25     42-0     43-0
BA26     42-0     43-0
BA27     42-0     43-0
BA28     42-0     43-0
BA29     42-0     43-0
BAD.LN   52-0
BADRH    51-0
BADRL    51-0
BAR      18-0     76-0     88-172   95-311   96-316   96-321   98-327   98-332   98-344  105-406  106-417  108-431  109-450  113-477
        113-503  119-535  121-711  121-737  122-755  123-767  123-771  123-772  123-777  123-1000 124-1004 126-1034 126-1035 126-1036
        129-1055 130-1061 130-1064 130-1067 130-1070 130-1077 130-1100 130-1102 130-1106 131-1116 131-1121 131-1122 131-1127 131-1131
        132-1142 132-1145 132-1147 132-1150 132-1151 132-1152 132-1154 132-1156 132-1157 132-1161 136-1166 136-1170 136-1171 136-1172
        136-1173 136-1175 136-1176 136-1203 137-1215 141-1273 141-1276 142-1321 142-1326 142-1327 142-1330 146-1347 146-1354 148-1372
        148-1375 148-1376 148-1405 148-1411 148-1414 148-1416 148-1423 148-1426 148-1427 148-1431 149-1445 149-1450 149-1460 150-1473
        150-1476 150-1477 150-1500 150-1506 150-1510 150-1517 151-1526 152-1543 153-1546 153-1553 153-1574 153-1577 153-1600 154-1610
        154-1611 154-1613 154-1614 154-1620 154-1621 154-1631 154-1633 154-1636 154-1651 155-1653 155-1660 155-1661 155-1667 155-1675
        155-1700 155-1701 156-1715 156-1716 156-1721 156-1723 156-1726 157-1735 157-1744 157-1746 157-1747 157-1757 157-1773 158-1775
        158-2004 159-2016 159-2023 159-2025 159-2032 159-2034 160-2035 160-2037 160-2040 160-2043 160-2044 160-2045 160-2047 160-2053
        160-2064 161-2074 161-2105 161-2106 162-2112 162-2117 162-2121 163-2134 163-2136 163-2144 163-2146 164-2150 164-2157 164-2161
        164-2163 164-2167 164-2175 164-2200 164-2204 164-2210 164-2213 164-2215 164-2220 165-2223 165-2225 165-2230 165-2234 165-2235
        165-2237 165-2240 165-2244 165-2253 165-2255 165-2260 165-2261 165-2262 165-2263 165-2264 165-2265 165-2266 166-2272 166-2276
        166-2306 166-2311 166-2314 167-2343 168-2344 168-2345 168-2347 169-2352 169-2362 169-2364 169-2372 169-2373 169-2401 169-2403
        169-2406 169-2412 169-2413 169-2417 169-2422 171-2425 171-2430 171-2431 171-2432 171-2433 171-2435 171-2442 171-2445 171-2455
        171-2460 171-2461 171-2462 171-2463 171-2464 172-2465 172-2467 172-2470 172-2471 172-2473 172-2474 173-2475 173-2476 173-2502
        173-2503 173-2505 176-2544 176-2546 176-2547 177-2551 177-2553 177-2564 178-2571 178-2573 178-2575 179-2603 179-2607 179-2614
        179-2616 179-2621 179-2626 179-2630 179-2632 180-2634 180-2636 180-2640 180-2647 180-2651 180-2652 180-2653 180-2662 180-2664
        180-2666 180-2667 181-2700 181-2702 181-2704 181-2712 181-2722 181-2726 181-2733 181-2740 182-2742 182-2757 182-2760 182-2762
        182-2764 182-2767 182-2774 182-2776 182-3005 182-3010 182-3013 184-3014 184-3021 185-3053 185-3054 185-3061 186-3071 186-3072
        187-3075 187-3116 187-3117 187-3121 187-3123 187-3125 187-3127 187-3131 187-3141 187-3143 187-3144 188-3147 188-3153 188-3160
        188-3161 188-3164 188-3171 188-3202 188-3204 189-3211 189-3212 189-3214 189-3215 189-3216 189-3220 189-3226 189-3227 189-3230
        189-3234 189-3237 189-3240 189-3241 189-3242 190-3247 190-3257 190-3272 192-3273 192-3275 192-3301 192-3302 192-3304 192-3306
        192-3312 192-3316 192-3317 192-3321 193-3325 193-3326 193-3331 193-3336 193-3337 194-3346 194-3351 194-3353 194-3354 194-3357
        194-3365 195-3371 195-3374 195-3375 195-3376 195-3377 195-3400 195-3401 196-3411 196-3417 196-3422 196-3425 196-3427 196-3431
        196-3435 196-3443 197-3457 198-3464 199-3474 199-3475 200-3500 200-3501 200-3506 201-3510 201-3511 201-3512 201-3513 201-3514
        201-3515 201-3516 201-3520 201-3523 203-3544 206-3556 209-3571 217-3631 217-3645 218-3652 218-3653 219-3677 219-3703 219-3715
        220-3717 220-3742 220-3743 220-3744 220-3746 220-3751 221-3752 221-3755 221-3757 221-3761 221-3762 221-3763 221-3786 221-3770
        221-3773 221-4004 221-4007 221-4013 221-4014 221-4016 221-4017 226-7641 227-7712 228-7742 228-7744 228-7747 228-7753
BCAID    18-0
BCAIS    18-0
BCGRP    64-0
BCICSR   41-0
```

```
BDCST   40-0
BDSNF   64-0    167-2341
BECC    64-0    182-2761 193-3327 193-3340
BECER   64-0
BERDN   64-0    123-773  148-1430 165-2247
BERMAX  67-0     67-0
BFRQ    59-0    122-761  194-3345
BFSV    59-0    154-1623 157-1763 163-2125 164-2164 165-2251 166-2275 169-2376 171-2441 173-2500 179-2602 179-2631 182-2771 194-3345
        194-3363
BFULL   64-0    154-1622 164-2212 165-2227 165-2242 166-2274 168-2350 173-2501 181-2737 182-2763 182-2773 193-3333 193-3334 194-3352
BGOOD   64-0    169-2360 169-2361 187-3120 188-3156
BGRUP   64-0    163-2134 166-2312 166-2322 176-2543 179-2614 193-3330 193-3341
BIBAD   13-0
BIBER   41-0
BICSR   41-0
BICSRE  43-0
BIDTYP  41-0
BIEADR  41-0
BIECSR  41-0
BIGPR0  41-0
BIGPR1  41-0
BIGPR2  41-0
BIGPR3  41-0
BIIDES  41-0
BIIMSK  41-0
BIPSDE  41-0
BIPSFC  41-0
BIPSRC  41-0
BIRDCM  40-0
BIRTRY  51-0
BISADR  41-0
BIT00   18-0     19-0     24-0     25-0     30-0     36-0     39-0     45-0     53-0     58-0     58-0     58-0     58-0     58-0
        59-0     59-0     59-0     60-0     61-0     63-0     64-0     65-0     69-0     69-0     88-167   91-251   121-705  121-714
        122-740  130-1101 137-1214 142-1320 148-1374 154-1634 154-1634 154-1634 155-1662 155-1701 155-1701 155-1701 156-1707 156-1733
        157-1747 157-1747 157-1747 163-2125 163-2134 164-2173 166-2303 166-2310 166-2312 174-2514 179-2602 179-2614 181-2735 182-2771
        206-3560
BIT01   18-0     24-0     25-0     39-0     45-0     53-0     58-0     58-0     59-0     59-0     60-0     61-0     63-0     64-0
        70-0
BIT02   18-0     24-0     39-0     58-0     58-0     59-0     60-0     63-0     64-0     69-0     202-3532 202-3534
BIT03   18-0     24-0     30-0     39-0     43-0     59-0     61-0     62-0     64-0     69-0     202-3532 202-3535
BIT04   18-0     22-0     24-0     25-0     30-0     39-0     49-0     58-0     58-0     59-0     60-0     61-0     62-0     64-0
        70-0
BIT05   18-0     22-0     24-0     24-0     25-0     30-0     39-0     58-0     59-0     59-0     60-0     61-0     62-0     64-0
        70-0     73-0
BIT06   18-0     21-0     22-0     24-0     24-0     25-0     25-0     30-0     39-0     49-0     58-0     58-0     59-0     61-0
        62-0     73-0
BIT07   18-0     22-0     24-0     30-0     35-0     35-0     39-0     43-0     50-0     58-0     58-0     59-0     61-0     62-0
        63-0     63-0     69-0     202-3532 202-3536
BIT08   18-0     21-0     22-0     39-0     43-0     46-0     50-0     58-0     59-0     62-0     62-0     64-0     73-0     73-0
        73-0     202-3533 202-3537
BIT09   18-0     19-0     39-0     50-0     58-0     59-0     62-0     62-0     64-0     73-0     73-0
BIT10   18-0     21-0     22-0     39-0     42-0     58-0     59-0     60-0     62-0     62-0     63-0     64-0     152-1544
BIT11   18-0     19-0     21-0     22-0     35-0     39-0     42-0     45-0     58-0     59-0     60-0     62-0     62-0     62-0
        64-0     152-1544
```



```
BIT12   18-0     19-0     22-0     35-0     39-0     42-0     56-0     58-0     59-0     60-0     61-0     62-0     62-0     63-0
        64-0     119-541  224-7400 226-7416
BIT13   18-0     19-0     22-0     35-0     39-0     43-0     56-0     58-0     58-0     59-0     59-0     60-0     61-0     62-0
        62-0     63-0     64-0
BIT14   18-0     19-0     22-0     35-0     35-0     39-0     42-0     43-0     53-0     58-0     59-0     59-0     60-0     61-0
        62-0     62-0     64-0     64-0     98-336   165-2256 165-2256 165-2256 195-3372 196-3432
BIT15   18-0     19-0     22-0     25-0     35-0     35-0     39-0     42-0     43-0     45-0     45-0     45-0     53-0     58-0
        59-0     59-0     60-0     61-0     61-0     61-0     61-0     62-0     62-0     63-0     64-0     64-0     73-0     88-166
        88-202   88-212   98-336   119-541  131-1123 136-1171 136-1204 150-1501 150-1507 151-1533 155-1674 155-1676 156-1724 158-1774
        160-2040 163-2144 165-2224 165-2242 165-2254 165-2267 166-2316 167-2334 169-2402 171-2443 174-2515 175-2520 176-2540 178-2566
        179-2624 181-2736 182-2764 194-3363 208-3566 224-7400 226-7416
BIT16   39-0     42-0     43-0     43-0     43-0
BIT17   39-0     42-0     43-0     43-0
BIT18   39-0     42-0     42-0     43-0
BIT19   39-0     42-0     42-0     43-0
BIT20   39-0     42-0     42-0     43-0
BIT21   39-0     42-0     42-0     43-0
BIT22   39-0     42-0     42-0     43-0
BIT23   39-0     42-0     42-0     43-0
BIT24   39-0     42-0     42-0     43-0
BIT25   39-0     42-0     42-0     43-0
BIT26   39-0     42-0     42-0     43-0
BIT27   39-0     42-0     42-0     43-0
BIT28   39-0     42-0     42-0     43-0
BIT29   39-0     42-0     42-0     43-0     44-0
BIT30   39-0     42-0     43-0     44-0     44-0
BIT31   39-0     43-0     44-0     44-0
BIUCSR  41-0
BIWRCM  40-0
BIWSTA  41-0
BLAST   64-0    165-2224 165-2242 165-2242 182-2764 182-3001 192-3274 194-3363
BLOCK   19-0
BLRWR   64-0    167-2342
BLSTB   64-0
BMAPDN  64-0
BNXCOP  64-0    169-2423
BOARD2  73-0     80-40
BRANCH 217-3634 221-3752
BRARS   64-0    165-2246 169-2404
BRCTS   64-0    165-2245 165-2250 169-2404 194-3347
BREG    18-0
BRLV4   43-0
BRLV4R  43-0
BRLV4S  43-0
BRLV5   43-0
BRLV5R  43-0
BRLV5S  43-0
BRLV6   43-0
BRLV6R  43-0
BRLV6S  43-0
BRLV7   43-0
BRLV7R  43-0
BRLV7S  43-0
BROKE   42-0
```

```
BRTRY   64-0
BTCNT   51-0      51-0     165-2225
BTROTH  45-0
BUF      18-0      18-0      18-0      95-314    98-330    98-331    98-333   113-500   113-505   119-536   121-713   121-721   121-726   123-1001
        130-1065  130-1072  130-1103  131-1123  132-1143  132-1146  137-1217  142-1331  146-1355  148-1373  148-1412  148-1424  148-1430  149-1452
        150-1472  150-1474  150-1501  150-1507  150-1520  151-1527  153-1554  153-1555  153-1561  153-1573  153-1576  154-1612  154-1622  154-1637
        154-1643  154-1646  154-1650  155-1671  155-1676  157-1750  158-1774  159-2012  159-2014  159-2017  159-2031  160-2046  160-2054  160-2060
        160-2070  162-2111  162-2122  163-2125  163-2133  163-2140  163-2147  164-2151  164-2160  164-2162  164-2165  164-2171  164-2173  164-2202
        164-2206  164-2212  164-2216  164-2221  165-2224  165-2250  166-2273  166-2304  166-2307  166-2310  166-2321  166-2322  168-2346  168-2350
        169-2361  169-2423  171-2451  173-2504  173-2506  176-2542  176-2543  177-2552  177-2561  178-2566  178-2572  178-2574  179-2602  179-2613
        179-2620  179-2627  179-2631  180-2633  180-2637  180-2644  180-2650  180-2657  180-2663  180-2665  181-2674  181-2675  181-2677  181-2701
        181-2703  181-2710  181-2714  181-2720  181-2724  181-2730  181-2737  182-2751  182-2755  182-2761  182-2763  182-2765  182-2770  182-2771
        182-3006  182-3011  184-3024  184-3025  185-3027  185-3031  185-3033  185-3035  185-3037  185-3041  185-3043  185-3045  185-3056  187-3120
        188-3156  188-3203  188-3205  189-3231  190-3250  192-3274  192-3322  193-3330  193-3333  193-3334  193-3341  194-3352  195-3367  196-3415
        196-3420  196-3423  196-3426  196-3430  196-3433  196-3437  196-3445  197-3460  199-3471  199-3476  201-3517  201-3522  201-3525  203-3541
        203-3547  212-3602  212-3603  212-3604  212-3605  212-3606  212-3607  212-3610  212-3611  212-3612  213-3614  213-3615  213-3616  213-3617
        213-3620  213-3621  213-3622  214-3623  214-3624  214-3625  217-3633  217-3644  218-3654  218-3660  218-3670  218-3673  219-3700  219-3704
        220-3716  220-3723  220-3724  220-3730  220-3731  220-3733  220-3735  220-3736  220-3740  220-3747  220-3750  221-3760  221-3771  221-4003
        226-7643  226-7644  226-7645  226-7646  226-7647  226-7650  226-7671  227-7723  228-7765  228-7765
BUF.56  65-0      65-0
BUF.BC  65-0      65-0      65-0
BUF.BP  64-0      64-0     131-1121  164-2204  164-2215  169-2406  172-2473  179-2607  179-2611  179-2621  180-2643  180-2665  180-2671  182-3013
        186-3071  189-3227
BUF.DL  65-0      65-0      65-0
BUF.EC  65-0      65-0
BUF.ED  65-0      65-0
BUF.GP  65-0      65-0     176-2544
BUF.HH  64-0      64-0     130-1064  130-1102  165-2255  169-2417  171-2464  185-3054  187-3075  187-3131  189-3230
BUF.HL  64-0      64-0     165-2262  171-2462  185-3053  188-3164  189-3226
BUF.LL  65-0      65-0      65-0      65-0
BUF.NL  64-0      64-0     164-2175  164-2200  164-2210  165-2223  165-2264  166-2272  168-2347  173-2476  181-2704  181-2712  181-2733  182-2762
        192-3273  193-3331  194-3351
BUF.SD  64-0      64-0     164-2157  165-2234  167-2343  169-2362  171-2430  172-2467  176-2546  179-2626  180-2651  180-2666  188-3160  188-3202
BUF.ST  64-0      64-0     123-772   148-1427  163-2136  165-2230  165-2244  166-2311  166-2314  168-2345  169-2352  169-2403  169-2422  172-2474
        179-2616  182-2760  182-2774  187-3117  188-3153  193-3326  193-3337  194-3346
BUF.TA  64-0      64-0     130-1070  163-2134  163-2144  179-2614  179-2632  192-3301  192-3306  193-3325
BUF.U1  65-0      65-0
BUF.U2  65-0      65-0      65-0
BUF.UA  64-0      65-0
BUF.US  65-0      65-0      65-0
BUFBEG  65-0      65-0
BUFEND  65-0      65-0
BUFLMT  45-0
BUFP00  49-0      49-0
BUFP01  49-0      49-0
BUFP02  49-0      49-0
BUFP03  49-0      49-0
BUFP04  49-0      49-0
BUFP05  49-0      49-0
BUFP06  49-0      49-0
BUFP07  49-0      49-0
BUFP08  49-0      49-0
BUFP09  49-0      49-0
BUFP10  49-0      49-0
```



```
BUFP11  49-0      49-0
BUFP12  49-0      49-0
BUFP13  49-0      49-0
BUFP14  49-0      49-0
BUFP15  50-0      50-0
BUFP16  50-0      50-0
BUFP17  50-0      50-0
BUFP18  50-0      50-0
BUFP19  50-0      50-0
BUFP20  50-0      50-0
BUFP21  50-0      50-0
BUFP22  50-0      50-0
BUFP23  50-0      50-0
BUFP24  50-0      50-0
BUFP25  50-0      50-0
BUFP26  50-0      50-0
BUFP27  50-0      50-0
BUFP28  51-0      51-0
BUFP29  51-0      51-0
BUFP30  51-0      51-0
BUFP31  51-0      51-0
BUFP32  51-0      51-0
BUFP33  51-0      51-0
BUFP34  51-0      51-0
BUFP35  51-0      51-0
BUFP36  51-0      51-0
BUFP37  51-0      51-0
BUFP38  51-0      51-0
BUFP39  51-0      51-0
BUFP40  51-0      51-0
BUFP41  51-0      51-0
BUFP42  51-0      51-0
BUFP43  51-0      51-0
BUFP44  51-0      51-0
BUFP45  51-0      51-0      51-0      51-0
BUFP46  51-0      51-0
BUFP47  51-0      51-0      51-0      51-0
BUFP48  51-0      51-0
BUFP49  51-0      51-0
BUFP50  51-0      51-0      51-0
BUFP51  51-0      51-0
BUFP52  51-0      52-0
BUFPTR  49-0      49-0
BUFR1.  65-0      65-0      65-0
BUFR2.  65-0
C$TYPE   1-0       1-0       8-0       8-0       8-0      12-0      12-0      12-0      13-0      13-0      13-0      14-0      14-0      15-0
        15-0      15-0      15-0      15-0      15-0      18-0      18-0      18-0      20-0      21-0      22-0      22-0      22-0      25-0
        27-0      29-0      30-0      30-0      38-0      45-0      45-0      45-0      46-0      46-0      46-0      51-0      51-0      51-0
        51-0      59-0      63-0      64-0      65-0      65-0      65-0      65-0      65-0      67-0      67-0      70-0      71-0      71-0
        74-0      74-0      74-0      74-0      74-0      75-0      76-0      76-3      76-4      79-31     80-44     83-73     83-103    83-104
        83-113    83-113    83-114    85-142    91-251    91-251    91-253    93-266    94-302    99-353    100-362   101-373   111-467   116-521
       117-527   119-541   121-701   176-2544  185-3055  202-3531  224-7400  224-7404  225-7416  226-7416  228-7760
C.AONL 153-1603  154-1624
C.ATTN 154-1624  154-1626  154-1630  154-1630  155-1664
```

LSCS FORM=QUAD

```
C.DCON  155-1665  227-7716
C.DUPL  152-1541  227-7731
C.DUPX  153-1604  154-1627
C.IMFF  154-1613  227-7701
C.LOGA  154-1634  154-1636
C.LOGS  149-1463  154-1633  157-1753  171-2436
C.SEEK  155-1673  227-7704
C.SUBC  153-1602  155-1660
C.SUBU  154-1646  228-7742
CCLASS   45-0
CCSRWO   48-0      48-0
CCSRW1   48-0      48-0      48-0      48-0
CDONE    12-0
CDUPLA  152-1543  153-1565
CDUPLB  153-1550
CDUPLC  153-1555  153-1562
CDUPLD  153-1551  153-1552  153-1563
CDUPLE  153-1545  153-1566
CF.ATN   50-0      50-0
CF.MSC   49-0      50-0     154-1634
CF.THS   49-0      50-0     154-1635
CFLAGS   50-0
CHGFLG   69-0     147-1367
CHGMOD   69-0     227-7675  227-7714
CKSTO   217-3644  217-3646  219-3701  219-3705  219-3710  219-3713
CLIMIT   49-0      49-0
CLK     107-425   109-447   111-455   111-456   111-457   111-460   111-461   111-462   111-464   111-471   111-472
CLKOO   111-472   114-514
CLKO1   111-464   113-505
CLKO2   113-504   113-505   114-514   114-514   115-514
CLK1    106-412   111-462
CLK10   111-461   113-476   113-501   119-536
CLK16   109-442   111-460   113-502
CLK32   111-456
CLK6    111-457
CLK64   111-455
CLKP    106-411   111-463   111-473   112-475
CLRERR   87-165
CLRTST   81-51
CMDCOF   47-0      47-0
CMDLEN   47-0      47-0      48-0      48-0
CMDLIM   45-0      52-0
CMDO     48-0
CMDPOF   47-0      48-0
CMDPTR   47-0      47-0
CMODEL   45-0      45-0      45-0
CMPBUF   51-0      51-0
CN.ERR   19-0     226-7640
CNT.LN   52-0
CNTFLG   49-0      50-0      50-0     154-1636
CNVTV1   48-0      48-0
CNVTV2   48-0      48-0
CNVTV3   48-0      48-0
CNVTV4   48-0      48-0
```



```
CODVER   46-0      46-0      46-0
COMPLT   69-0     156-1711
CON.ER  228-7760  228-7774  229-8000
CON.ST   50-0
CONT     24-0     124-1006  125-1022  125-1024  140-1247  140-1253  140-1253  140-1257  140-1270  156-1720  163-2132  163-2141  164-2207  179-2612
        180-2641  180-2642  180-2654  180-2655  181-2732  188-3200  188-3201  196-3410  196-3446
CONT.F  143-1333  145-1344  146-1356
CONTCD   69-0     141-1305  142-1314  143-1336
COPY4     1-0     185-3047
CPE      12-0      12-0      12-0
CR       18-0      18-0      18-0
CRDY     24-0     140-1247
CRI      17-0      76-4      79-34     80-41     80-46     82-53     82-60     83-67     83-73     83-101    83-102    83-106    83-107    83-110
         84-114    84-115    86-153    86-154    89-237    93-274    94-276    95-310    96-315    96-317    96-320   100-354   108-435   108-437
        111-465   111-466   111-467   111-470   114-511   114-512   117-525   117-527
CRY      13-0      88-174   188-3155  189-3223  190-3262  190-3265  190-3267  208-3566  209-3571  218-3660  220-3740
CSEEKA  151-1533  151-1535  155-1675
CSERR    69-0     142-1325
CSR      14-0      99-351   121-735   124-1016  125-1031  156-1717  157-1766  202-3534
CTMOUT   46-0
CYCLE    36-0
CYLSTR   45-0
D.ALBF  169-2406  179-2610  182-3006
D.ARTC  180-2671  186-3064  188-3200
D.ATTN  126-1053  127-1053
D.AVAL  126-1047  128-1054
D.BFCA  123-1002  173-2476
D.BFCK  130-1062  163-2123  173-2475  179-2600
D.CLCS  137-1207  140-1270  171-2424  185-3050  196-3454
D.CLRS  123-776   148-1421  150-1504  154-1622  159-2011  160-2071  161-2102  165-2270  171-2451
D.CLUR  166-2273  166-2306  194-3355
D.CPUL  130-1066  130-1105  130-1110  174-2507  175-2522  175-2523  175-2525  175-2526  175-2531  182-2756  188-3176  195-3403  205-3554
D.DBNO  188-3150  189-3215  190-3254
D.DIAG   82-53
D.DIVD  190-3257
D.DNSK  149-1467  150-1522  151-1527  151-1537  157-1771  159-2016
D.DONE  154-1623  159-2022  160-2042  161-2103  227-7707
D.DONZ  155-1703  157-1765  159-2015
D.DTER  167-2341  181-2676
D.DVDA  190-3261  190-3267
D.DVDB  190-3262  190-3270
D.DVDC  190-3262  190-3264
D.DVDD  190-3263  190-3265
D.DVDE  190-3267  190-3271
D.DVDF  190-3265  190-3271
D.DVDG  190-3266  190-3272
D.E.C.    1-0
D.ECKA  156-1717  156-1721  157-1766
D.ECKB  149-1452  156-1705  156-1710  156-1713  156-1723
D.ECKC  156-1724  156-1730
D.ECKD  156-1733  157-1735
D.ECKR  149-1437  150-1503  151-1540  156-1704
D.ECKS  145-1346  148-1400  150-1523  151-1531  156-1714
D.ECRA  156-1704  156-1711
```

```
D.ECRB  149-1440 149-1453 156-1712
D.EERR  167-2340 167-2341 167-2343
D.END    82-55    82-61    83-65    83-71    83-76    83-102   84-114   84-131
D.ER    228-7770 228-7774 228-7776 229-8000
D.EROA  158-1774 158-2006
D.ERIA  123-770  123-774  123-776  136-1205
D.ERCK  149-1451 155-1673 155-1674 156-1706 156-1707 156-1715
D.ERRO  157-1773 226-7661
D.ERR1  122-767  123-767
D.ERRD  123-1002 193-3334
D.GECC  119-535  182-2744 182-2745 182-2751 182-3002
D.GGRP  171-2443 176-2535 176-2544 176-2545
D.GMST  121-725  153-1566 154-1641 154-1646 159-2011 159-2013 160-2065 162-2110 166-2302 173-2500 209-3572
D.GPRE  173-2477 173-2502
D.GRPC  163-2137 176-2537 179-2617
D.GSCB  159-2032 228-7737
D.GUNF  147-1360 154-1652 159-2023
D.HCEO  169-2357 169-2361
D.HCEA  169-2371 169-2376 169-2401
D.HCEC  169-2402 169-2415
D.HCED  169-2400 169-2422 171-2447
D.HCER  163-2143 169-2352 179-2623
D.IDLA  121-700  121-702
D.IDLB  121-707  121-710
D.IDLC  121-727  121-730
D.IDLD  121-717  121-732
D.IDLE   84-135  121-701  121-701  121-702  121-716  121-720  121-724  121-727  121-735  122-741  122-744  122-746  122-750  122-757
        122-760  122-766  126-1037 136-1200 166-2305 166-2312 169-2354 169-2407 171-2457 176-2543 179-2604 180-2672 192-3322 206-3565
D.IDLF  121-732  121-736
D.IDLG  121-723  121-731  122-742
D.IDLH  122-740  122-743
D.IDLI  121-734  121-736  122-745  122-747  122-751
D.IDLJ  121-715  122-747
D.IDLM  121-714  122-742  122-752
D.IDLN  122-753  122-755
D.INIT  121-721  125-1022 149-1465 204-3551
D.INSO  155-1676 158-2000 158-2001 158-2003 158-2005
D.INS1  149-1462 149-1464 149-1466 150-1524 153-1607 154-1615 154-1616 154-1632 155-1662 155-1663 156-1712 156-1714 158-2007 160-2036
        160-2036 227-7723
D.INS2  148-1374 148-1401 150-1472 155-1672 155-1672
D.INSV  150-1513 154-1612 155-1657 155-1672 158-2005 158-2006 158-2010 160-2036 160-2037 226-7653 226-7664
D.IODA  165-2242 165-2243 165-2251
D.IODB  165-2252 169-2421 169-2423
D.IODC  165-2251 165-2254 165-2265
D.IODD  165-2267 166-2272
D.IODN  165-2226 165-2231 168-2351 182-2775
D.IOEA  171-2453 171-2455
D.IOER  167-2326 167-2332 167-2337 169-2355 171-2424
D.IOEX  132-1144 132-1153 132-1154 163-2125 166-2300 166-2303 166-2305 166-2310 166-2322 179-2602
D.IOPR  130-1063 172-2467 179-2612
D.IOPW  163-2132 172-2465
D.LDHD  131-1123 169-2363 171-2437 184-3020 185-3053 188-3151 192-3303
D.LOGF  131-1136 154-1645 157-1756 160-2043 165-2252
D.LRER  167-2342 179-2624 181-2736
```



```
D.NCLK  156-1720 157-1766 157-1767
D.NOBF  163-2124 166-2310 179-2601
D.OFFD  122-745  148-1412 157-1763 160-2045
D.OFFL  154-1625 160-2051 160-2064
D.OLRD  160-2071 227-7711
D.ONLN  126-1046 129-1055
D.OVER  164-2204 167-2323 181-2732
D.PCMB  126-1041 126-1045
D.PCMD  122-762  126-1034
D.RAMP  228-7765 228-7775 229-8000
D.RAMX  121-704
D.RBLH  165-2257 169-2411 171-2460
D.RCLR  137-1212 140-1271
D.RCTE  167-2340 169-2405
D.READ  126-1043 160-2072 179-2600
D.RECC  182-2744 182-2755
D.RED1  179-2625 179-2626 179-2632
D.REDA  123-1003 179-2602
D.REDB  179-2613 182-2774
D.REDC  179-2614 179-2620 179-2632
D.REDE  179-2621 180-2633
D.REDF  180-2652 180-2667 181-2673 181-2673
D.REDG  181-2677 181-2677 181-2700
D.REDH  181-2674 181-2675 181-2700
D.REDI  181-2701 181-2701
D.REDJ  181-2737 182-2764
D.REDL  182-2767 182-2773
D.REDM  182-2775 182-3001
D.REDN  182-2764 182-2776
D.REDX  179-2606 198-3467
D.RLF1  180-2637 180-2653
D.RLF2  180-2663 180-2671
D.RNBF  179-2611 180-2640 180-2664 180-2670
D.RPLC  126-1044 130-1061
D.RRDE  163-2136 166-2314 167-2335 169-2356 179-2616
D.RRDY  105-406  117-525
D.RRWE  163-2131 164-2212 167-2333 179-2607
D.RSAV  192-3276 194-3342 194-3365
D.RSTR   76-2     76-3     84-134
D.RTDE  164-2206 167-2327 181-2733
D.RV2A  192-3317 193-3335 193-3341
D.RV4A  165-2271 193-3335
D.RV4B  193-3324 193-3337
D.RV8A  194-3350 194-3353
D.RVC2  192-3273 226-7643
D.RVC4  193-3323 226-7644 226-7645
D.RVC6  193-3323
D.RVC8  194-3342 226-7646 226-7647 226-7650
D.RVCA  132-1146 132-1151
D.RVCT  122-765  132-1142
D.RVDX  192-3321
D.RWRA  167-2323 167-2330 167-2332
D.RWRE  163-2145 167-2331 167-2334
D.SDIE  157-1742 157-1770
```

LSCS FORM=QUAD

```
D.SDIF  157-1770 157-1770 157-1772 158-1777
D.SDIG  150-1514 157-1771
D.SEEK  122-764  133-1163
D.SETA  166-2301 166-2303
D.SETB  182-2750 182-2753 182-3005
D.SETS  131-1141 136-1206 166-2302 169-2414 194-3361 194-3363 194-3364
D.SGRP  163-2135 166-2313 176-2535 179-2615
D.SLBA  189-3210 189-3234
D.SLBN  171-2440 189-3233
D.SMOV  149-1456 149-1457 161-2105 192-3300 194-3344
D.SMVX  161-2104 161-2107
D.SRTC  184-3014 185-3055 185-3056 196-3411
D.SSET  122-754  148-1415 148-1422 154-1630 159-2013
D.STOP  226-7642 228-7760 228-7761 228-7762 228-7763 228-7764 228-7771 228-7772 228-7773 228-7777
D.STPA  226-7416 226-7640 226-7642
D.SUCH  126-1051 134-1164
D.TIME  157-1755 162-2112
D.TIML  150-1501 162-2121
D.TIMO  167-2324 228-7766
D.TIMR  136-1167 150-1525 162-2110 171-2456
D.TKEA  167-2337
D.TKER  130-1076 167-2336 169-2361
D.TOER  167-2326 167-2326 167-2335
D.TOPO  126-1050 135-1165
D.VECA  136-1167 136-1201
D.VECB  136-1202 136-1204 136-1206
D.VECP  123-775  126-1052 127-1053 128-1054 129-1057 129-1060 133-1163 134-1164 135-1165 136-1166
D.VECT  122-763  136-1170
D.WAIT  161-2073 161-2073 227-7726
D.WDAT  164-2151 178-2570 178-2570 178-2571
D.WDTB  178-2572 178-2572
D.WDTC  178-2574 178-2574 178-2575
D.WDTD  178-2576 178-2576
D.WHDA  177-2552 177-2552
D.WHDB  177-2554 177-2554 177-2556
D.WHDC  177-2555 177-2555
D.WHDR  177-2550 177-2550 177-2551 196-3415
D.WRIT  126-1042 163-2123 193-3333
D.WRTB  163-2133 165-2230
D.WRTC  163-2134 163-2140 163-2144
D.WRTE  163-2141 163-2146
D.WRTF  164-2171 164-2176
D.WRTG  164-2175 164-2201
D.WRTI  164-2215 165-2223
D.WRTJ  165-2224 165-2227
D.WRTX  163-2130 198-3470
D.WSAF  163-2127 175-2520 179-2605
D.WSGA  174-2515 174-2516
D.WSIG  130-1071 163-2130 164-2211 174-2513 174-2514 179-2606 186-3067 196-3405 196-3450
D.ZATT  148-1407 153-1605 161-2074 161-2074 227-7720 227-7734 228-7743
DATEND   65-0     65-0     66-0
DATT     59-0    122-754  153-1561 160-2060 161-2102
DBAR     17-0    105-406  106-417  108-431  109-450  113-477  113-503  119-535  121-706  121-710  121-711  121-737  122-755  123-767
        123-771  123-777  123-1000 126-1035 129-1055 130-1077 132-1142 132-1146 132-1151 132-1153 136-1166 136-1170 136-1171 136-1172
```



```
        136-1173 136-1174 136-1175 136-1176 136-1203 148-1405 148-1411 148-1423 148-1426 148-1431 149-1433 149-1454 149-1460 150-1473
        150-1500 150-1506 150-1510 151-1526 152-1543 153-1551 153-1574 153-1577 153-1600 154-1610 154-1611 154-1613 154-1620 154-1631
        154-1633 154-1651 155-1653 155-1660 155-1661 155-1667 155-1675 155-1700 155-1701 156-1715 156-1721 156-1723 156-1726 157-1735
        157-1744 157-1746 157-1757 157-1773 158-1775 158-2004 159-2023 159-2025 159-2032 159-2034 160-2035 160-2037 160-2040 160-2043
        160-2044 160-2045 160-2047 160-2064 161-2074 162-2112 162-2117 162-2121 164-2160 164-2161 164-2166 164-2167 164-2206 165-2235
        165-2237 165-2240 165-2253 165-2260 165-2265 165-2286 166-2276 166-2306 168-2344 169-2364 169-2372 169-2373 169-2401 169-2412
        169-2413 171-2432 171-2433 171-2435 171-2442 171-2445 171-2455 171-2460 171-2461 171-2463 173-2475 173-2502 178-2573 178-2575
        178-2575 179-2603 179-2627 179-2630 180-2633 180-2637 180-2644 180-2652 180-2663 180-2665 180-2667 181-2700 181-2700 181-2701
        181-2702 182-3005 182-3005 188-3171 188-3173 188-3203 188-3204 188-3207 189-3211 189-3234 189-3237 189-3240 194-3357 194-3365
        196-3423 196-3424 196-3451 209-3571 227-7702 227-7712 228-7742
DCERR    69-0     142-1320 149-1451 156-1705 156-1710 156-1722 156-1733 157-1741 200-3503
DCLASS   45-0
DCLK     14-0     137-1226 137-1227 137-1230 137-1231 137-1232 137-1233 137-1234 137-1235 137-1236 137-1237 138-1240 138-1241 138-1242
        140-1251 140-1252 140-1254 140-1261 140-1263 140-1264 181-2674 181-2675 184-3024 184-3025
DCMASK   62-0
DCN.ST   69-0     148-1410 148-1417 148-1420 148-1425 155-1666
DCN.TT   69-0     148-1404
DCRD      18-0      18-0      18-0      76-2      76-4      77-14     80-46     82-53     82-60     83-67     83-101    83-106    84-115    84-135
          86-153    86-154    91-252    93-274    94-276    95-310    96-315    96-320   100-354   108-437   111-465   111-466   111-467   111-470
        114-511   114-512   117-525   117-527   124-1011 228-7770 228-7776
DCRS      18-0      18-0      18-0      91-253   121-703   124-1007
DCRTST   83-64     91-251
DDC      22-0     111-466   111-467   114-511   114-512   117-525   117-527
DDD      22-0     111-465   111-470
DER      17-0      79-32     79-33     80-35     80-36     80-37     80-40     80-42     80-43     80-44     80-47     82-56     82-62     83-66
          83-72     83-77     83-111    83-112    84-132    86-152    87-165   116-524  224-7404 224-7405 224-7406 224-7407 224-7410 224-7411
        224-7412 224-7413 224-7414 224-7415
DERR     59-0     131-1137 136-1201 157-1765 158-1776 160-2066 165-2271 169-2377 169-2421 171-2444 193-3323 194-3343 194-3360
DEVCL    45-0
DFAIL    21-0      21-0      79-34     80-41
DHANG    80-46     80-50     84-132
DI       73-0
DIAG     73-0
DINITA  125-1024 125-1026
DINITB  125-1025 125-1027
DINITC  125-1030 125-1032
DISCON   69-0     148-1413
DISM     48-0
DISN     48-0
DL1144  223-4020
DL11AD  223-4020
DM.BEG   65-0      65-0      65-0     217-3626 217-3627 221-3772 221-3773 221-4006 221-4007
DMBEG    19-0     121-702   171-2426 206-3565 217-3632 221-4003 221-4003 221-4003 221-4003
DMBPC    45-0     221-3772 221-4003 221-4004 221-4006
DMDBG   216-3626 216-3626 217-3630 223-4020
DMDT    223-4020
DMNTRY  121-700  216-3626
DMODE    19-0     121-701  164-2214 165-2232 169-2354 179-2624 181-2741 182-2777 187-3100 187-3140 188-3155 189-3213 189-3235 197-3456
        206-3563
DMODT    45-0     221-4004
DMOVH    45-0
DMOVL    45-0
DMPC    217-3632
DMREGO   47-0      47-0     131-1122 171-2425 195-3371 195-3400 196-3417 196-3443 198-3464 200-3500 200-3506 201-3510 203-3544 228-7747
```

```
        228-7753
DMREG1   47-0      47-0     195-3401 197-3457 199-3474 200-3501
DMREG2   47-0      47-0     124-1004 163-2146 163-2147 180-2647 180-2653 196-3435 206-3556 228-7744
DMREG3   47-0      47-0     173-2505 185-3061
DMREG4   47-0      47-0     163-2147 172-2465 173-2503 177-2553 177-2553
DMREG5   47-0      47-0     130-1061 131-1127 171-2431 195-3377 196-3431
DMREG6   47-0      47-0
DMREG7   47-0      47-0
DMSTR    45-0     217-3626
DMTEMP   49-0      49-0
DOFFDA  160-2053 160-2063
DOFFDB  160-2056 160-2057 160-2061
DPF      14-0      82-56     82-62     83-66     83-72     83-77     83-112    87-162   100-354   111-473   113-502   153-1571 153-1572 171-2431
        188-3157 195-3377
DPROC    12-0
DRAMPE   83-67     95-304
DRAVL    59-0     121-714   121-716   148-1406 148-1415 148-1421 150-1504 150-1511 153-1552 153-1556 153-1566 153-1570 153-1572 154-1650
        157-1756 157-1760 157-1762 158-1777 160-2056 160-2065 160-2067
DRDUP    59-0     121-716   148-1407 148-1417 150-1512 153-1552 153-1560 153-1561 153-1566 153-1570 153-1571 153-1603 157-1756 157-1762
        160-2050 160-2055 160-2057 160-2065 160-2067
DRDY     25-0      99-353   100-364   137-1214 156-1707 174-2514 175-2520 181-2735
DRINIT   21-0      21-0
DROMPE   82-60     93-266
DRPLC1  131-1132 131-1141
DRPLCA  130-1067 130-1075
DRPLCB  130-1107 130-1111
DRPLCC  130-1107 130-1112
DRPLCD  131-1117 131-1127
DRPLCX  130-1105 131-1115
DRV.AT   63-0      63-0     159-2033 226-7671
DRV.AV   62-0      63-0     159-2033 226-7671 227-7723
DRV.C1   62-0     153-1604
DRV.C2   62-0
DRV.C3   62-0
DRV.C4   62-0
DRV.DB   62-0     147-1364 147-1365
DRV.DD   62-0
DRV.DE   62-0     155-1670 158-2001
DRV.DF   62-0      62-0     226-7657
DRV.DR   61-0     149-1461
DRV.EL   61-0     149-1461
DRV.FO   62-0     147-1365 147-1366
DRV.OA   61-0     148-1403 153-1602 161-2075
DRV.PE   62-0      62-0
DRV.PS   61-0     155-1665
DRV.RE   62-0      62-0
DRV.RR   61-0     150-1471 150-1472
DRV.RU   61-0     122-740   155-1662 155-1665 155-1666 155-1666
DRV.S1   62-0      62-0
DRV.S2   62-0      62-0
DRV.S3   62-0      62-0
DRV.S4   62-0      62-0
DRV.S7   62-0     147-1364 147-1365
DRV.SN   62-0     155-1662
```



```
DRV.SR   61-0     158-2000
DRV.SU   61-0      63-0     147-1360 147-1365 147-1370 154-1652 155-1654
DRV.U1   61-0      61-0     154-1612
DRV.U2   61-0      61-0
DRV.U3   61-0      61-0
DRV.U4   61-0      61-0     155-1654
DRV.UM   63-0     153-1550 153-1576 154-1612
DRV.W1   62-0
DRV.W2   62-0     147-1363
DRV.W3   62-0     147-1363
DRV.W4   62-0     147-1363
DRV.WE   62-0      62-0
DRVCLR   69-0     148-1401
DRVOL    59-0     121-714   121-716   121-717   148-1417 148-1422 150-1512 153-1552 153-1556 153-1566 153-1571 153-1601 154-1647 157-1760
        158-1777 160-2065
DSDIEA  157-1740 157-1742 157-1744 157-1745
DSDIEB  157-1761 157-1763
DSDIEC  157-1747 157-1753
DSER     14-0      99-350   116-523   121-716   122-750   137-1212 155-1673 161-2100 164-2206 181-2733 196-3447 202-3536 202-3537
DSK.LN   52-0
DSLEDS   82-57     86-151
DSR      14-0      99-352   100-360   202-3535
DSTSH    45-0
DSTSL    45-0
DTEMP1   50-0      50-0
DTIMEA  162-2113 162-2122
DTMP     17-0
DTSTBL  224-7404
DUP.LN   52-0
DUPVC    45-0
DXFC     19-0     121-700   206-3561 206-3562
ECC      18-0     110-454   113-505   181-2736 182-3002
ECC9.1   48-0      48-0      48-0
ECC9.2   48-0      48-0      48-0
ECC9.3   48-0      48-0
ECCMSK   73-0
ECCRD   108-430  108-432   109-440
ECHOCD   69-0     203-3542
ECOUNT   49-0      49-0
ECSUMH   73-0
ECSUML   73-0
EDSEED   45-0
EF.BBR   59-0
EF.BBU   59-0
EF.LOG   59-0
ELEV     61-0
ELOC     48-0
ELPM     48-0      48-0
ELPN     48-0
ELPO     48-0      48-0      48-0
ELPP     48-0      48-0      48-0      48-0      48-0      48-0
ENCACH   65-0      65-0
END.F   143-1337 145-1345 146-1354
ENDCD    69-0      69-0     141-1305 142-1315 142-1322 142-1324
```

```
EQ       13-0     47-0     48-0     49-0     59-0     65-0     67-0     76-5     78-22    90-243   90-247   121-714  122-740  122-767
        126-1040 126-1040 126-1040 126-1040 126-1040 126-1040 126-1053 130-1101 136-1171 136-1204 137-1214 140-1253 141-1305 141-1305
        142-1320 148-1374 148-1375 148-1417 149-1445 149-1450 149-1451 150-1502 151-1533 153-1607 154-1630 154-1632 154-1634 154-1634
        155-1662 155-1666 155-1672 155-1674 155-1676 155-1701 155-1701 156-1705 156-1707 156-1724 156-1733 157-1747 157-1747 157-1770
        158-1774 160-2036 160-2040 161-2074 163-2125 163-2134 163-2141 163-2144 163-2147 164-2173 164-2207 165-2224 165-2242 165-2254
        165-2256 165-2267 166-2303 166-2310 166-2312 166-2316 167-2326 167-2334 169-2402 171-2443 174-2514 174-2515 175-2520 176-2540
        177-2553 178-2566 179-2602 179-2614 179-2624 180-2643 180-2665 181-2735 181-2736 182-2764 182-2771 184-3021 187-3141 189-3211
        190-3255 192-3305 194-3363 196-3446 208-3566 221-4003 221-4003 221-4003 221-4003 223-4020 226-7643 228-7760 228-7760 228-7760
ER.BCA   67-0
ER.BP1   67-0
ER.DMX   67-0
ER.HTO   67-0
ER.INT   67-0
ER.IWR   67-0
ER.MER   67-0
ER.MRR   67-0
ER.MST   67-0
ER.NIM   67-0
ER.PRD   67-0
ER.PWR   67-0
ER.RAP   67-0     90-241   90-245   95-306
ER.ROP   67-0     93-267
ER.RP2   67-0
ER.RRD   67-0
ER.RTO   67-0     67-0     67-0
ER.RWR   67-0
ER.SAP   67-0     90-240   90-244   228-7775 228-7776
ER.SOP   67-0     228-7767 228-7770
ER.SRP   67-0     67-0
ER.STP   67-0
ER.TMO   67-0     93-272   167-2324
ER.VCI   67-0
ER.WP2   67-0
ERECOV   69-0     227-7705
ERR      35-0     38-0
ERR00    38-0     76-3     76-10    77-12
ERR01    38-0     79-32
ERR02    38-0     80-35
ERR03    38-0     80-36
ERR04    38-0     80-37
ERR05    38-0
ERR06    38-0
ERR07    38-0     38-0
ERRB1A   79-31    88-166   88-167   88-170   88-171   88-172   88-173   88-174   88-175   88-176   88-200   88-201   88-202   88-203
         88-207   88-210   88-212   88-213   89-235   89-237
ERRB1E   79-32    90-242   90-243   90-246   90-247   92-265   93-270   93-273   93-274   94-277
ERRB2    80-35    80-36    80-37    80-40    84-133
ERRB2E   80-35    87-162   87-163   91-256   95-307   96-321   98-331   98-334
ERRB2H   80-36    99-345   99-346   99-347   99-350   99-351   99-352   101-372  104-400  104-402  104-403  105-404  106-414  106-424
        107-427  108-431  108-433  108-435  111-463  111-466  116-517  116-520  116-522  116-523  117-525
ERRB2I   80-37
ERRBIT   73-0     80-42
ERREG    37-0
ERRINI   60-0     157-1734 158-2002
```



```
ERRINP   60-0     136-1171 155-1676 156-1724 156-1725 158-1774 160-2040
ERRIP    59-0     121-723  122-760  122-765  136-1202 136-1204 154-1647 165-2243 165-2257 194-3361
ERRRTC   45-0
ERRSET   79-34    80-42
ERRVEC   60-0
ETST1    12-0
EVAL     48-0
FAILUR   48-0     48-0     48-0     48-0     95-312   98-326   226-7641
FAIRCT   61-0
FCT.MD   45-0
FCT.V1   45-0     45-0
FCT.V2   45-0
FDCLK    22-0
FDIAG    61-0
FETCH   217-3631 218-3661 218-3672 218-3673 220-3723 220-3726 220-3731 220-3733 220-3734 220-3736 220-3740 220-3741 220-3747 221-3757
        221-4010
FETCH2  217-3632 221-3763 221-4013 221-4014 221-4016 221-4017
FETCH3  217-3634
FLAG     35-0
FLOAT    18-0
FM.BAD   58-0
FM.CNT   58-0
FM.DSK   58-0     165-2252 171-2436
FM.SDI   58-0     131-1136 149-1463 157-1753 157-1755
FORCE    43-0
FORICD   69-0     195-3404
FORSCD   69-0     131-1120 196-3434
FPE      24-0
FRERR    69-0     142-1313
FRMCD    45-0
FSEEK    60-0     150-1474 155-1671
FSTST    83-74    84-114
FTEST    12-0
GENSYN  104-375  106-410  109-441  117-533
GETCRY  217-3630 220-3725
GETECO  119-535  119-540
GETECC  108-434  119-537
GETRB   219-3700 219-3702 219-3711
GETRBX  217-3642 219-3703
GETSTA   69-0     226-7655 226-7667 227-7727
GO       36-0
GORD     15-0
GOWR     15-0
GPR0     43-0
GPR1     43-0
GPR2     43-0
GPR3     43-0
GSEL     59-0     150-1516 154-1617 155-1703 157-1764
GTCCHR   69-0     227-7735
GTCRSP   69-0     149-1435
GTE      13-0
GTUCHR   69-0     226-7673 228-7740
GTURSP   69-0     149-1444
H11COD   46-0     46-0     46-0
```

CROSS REFERENCE TABLE (CREF V01-08)

```
H11XOR  46-0    46-0    46-0    46-0    187-3113
HADD    15-0
HANG    17-0    76-5    76-5    80-45   85-136  85-137  85-141
HD.TO   82-53   224-7404
HD.T1   82-53   224-7405
HD.T2   82-53   224-7406
HD.T8   83-100  224-7414
HDCOD   45-0    130-1065 169-2367 169-2410 169-2420 171-2440 187-3076 187-3105 187-3133 187-3134 187-3142 189-3210 190-3254
HDLMT   49-0    49-0    164-2163 172-2471 181-2726 186-3072 187-3116 189-3212
HDRAD1  187-3134 187-3137
HDRANA  186-3066 187-3103
HDRANB  187-3105 187-3106 187-3121
HDRANC  187-3122 187-3125
HDRAND  187-3126 187-3131
HDRANE  187-3143 188-3150
HDRANL  130-1074 163-2142 179-2622 186-3064
HDRCCH  187-3125 187-3127 190-3245
HDRCCL  187-3121 187-3123 190-3243
HDRCHK  190-3244 190-3247 190-3252
HDRCKE  190-3247 190-3253
HDRCKH  187-3107 187-3111 187-3114 190-3246
HDRCKL  190-3244
HDRCMD  184-3023 184-3023
HDRCME  184-3022 184-3024
HDRCMF  185-3026 185-3050
HDRCMG  185-3052
HDRCMP  130-1073 163-2140 179-2620 184-3014
HDRNOM  186-3071 187-3115 187-3120 187-3124 187-3130 187-3137 188-3147 188-3155 188-3156 188-3173 188-3175
HDRTMO  45-0
HDSLB1  188-3204 188-3207
HDSUBA  187-3140 188-3204 189-3213
HEADER  47-0    47-0    47-0    47-0    149-1445 149-1450 149-1450 181-2740 182-2757 184-3021 187-3121 187-3123 187-3125 187-3127
        190-3244 190-3246 190-3257 190-3272
HES     42-0
HI.OFF  90-242  90-242  90-243
HI1     185-3031
HI2     185-3035
HI3     185-3041
HI4     185-3045
HIBYT   45-0    46-0    93-270  94-300  125-1023 125-1027 136-1177 142-1316 142-1324 147-1364 152-1541 211-3601 217-3627 218-3672
        221-3773 221-4002 221-4007
HINIB   45-0    149-1442
HIPAR   90-242  224-7403 228-7774 229-8000
HOSTF   45-0
HOSTMO  50-0    50-0    150-1476
HSTQUE  50-0    50-0
IDENT   40-0
IDX     25-0    99-353  101-370  130-1066 130-1105 131-1124 175-2521 182-2756 185-3063 188-3176 195-3402 195-3403 205-3554
IE      35-0
IMMED   217-3641 217-3645
INCRTN  90-250  93-275  95-310  98-322  98-342  108-437  116-514
INDIAG  19-0    82-53   82-60   83-67   83-110  84-114  89-237  93-274  94-276  96-315  226-7642 228-7766
INDRCT  218-3653 219-3715 220-3746 220-3750
INI     24-0    125-1022 125-1024
```


CROSS REFERENCE TABLE (CREF V01-08)

```
INIT1   132-1150 214-3624
INITM1  154-1651 165-2225 214-3625
INPLEN  47-0    149-1447 150-1502 151-1536
INPUT   47-0    148-1375 149-1446 149-1453 226-7660 226-7663 226-7666 226-7672 227-7676 227-7700 227-7703 227-7706 227-7713 227-7715
        227-7717 227-7722 227-7725 227-7733
INSEEK  69-0    150-1520
INSTR   36-0    36-0
INTI    35-0
INTR    40-0
INTVEC  49-0    49-0
INVAL   40-0
IOACC   44-0
IOCLK   60-0    167-2326 171-2452
IOCNT   60-0    171-2446 171-2450
IOMSK   45-0
IORTY   60-0    155-1676 158-1774 171-2447
IORWR   60-0    167-2332
IOSEK   60-0    158-2006 167-2337
IPINTR  40-0
IPREG   41-0
IRCI    40-0
IUAR    17-0
JCOND   221-3754 221-4011
JEQJPL  221-3764 221-4013 221-4015
JMPTST  76-2    76-5
JNEJMI  221-3771 221-4012 221-4016
JSBXFC  221-3755 221-3765
JSIGN   221-4011 221-4015
JUMP    76-7    77-15
L.BADO  57-0    57-0
L.BAD1  57-0
L.CHVR  57-0
L.CNTO  57-0    57-0
L.CNT1  57-0    57-0
L.CNT2  57-0    57-0
L.CNT3  57-0
L.CRFO  57-0    57-0
L.CRF1  57-0
L.CSVR  57-0    57-0
L.EVNT  57-0
L.FLGS  57-0
L.FMT   57-0
L.HDRO  57-0    57-0
L.HDR1  57-0
L.MLUN  57-0
L.RTRY  57-0
L.SDIO  57-0    57-0
L.SDI1  57-0    57-0
L.SDI2  57-0    57-0
L.SDI3  57-0    57-0
L.SDI4  57-0    58-0
L.SDI5  58-0
L.SEQ   57-0
L.UHVR  57-0
```

```
L.UNIT   57-0
L.UNTO   57-0      57-0
L.UNT1   57-0      57-0
L.UNT2   57-0      57-0
L.UNT3   57-0
L.USVR   57-0      57-0
L.VSE0   57-0      57-0
L.VSE1   57-0
LAB       1-0
LADD     15-0
LATE     14-0     164-2204  181-2732  196-3445
LBNMSK   45-0     187-3141  189-3214
LBNSTR   45-0     190-3255  190-3255
LBWR     15-0
LCOM     15-0
LDRT1   212-3610  219-3703  221-3766  221-4004
LED1     22-0      22-0      80-41     86-151
LED2     22-0      22-0      79-34
LED4     22-0      22-0      79-34     80-41
LED8     22-0      22-0
LEDS     22-0      77-14     82-60     83-67     83-110    84-114    84-134    89-237    93-274    94-276    96-315   100-354   108-437
LESS     12-0
LEVOR2  140-1251
LEVORA  140-1250  140-1250
LEVORB  140-1253  140-1255
LEVORC  140-1251  140-1252  140-1254  140-1257
LEVORD  140-1244  141-1273  141-1303  203-3544
LEVORE  140-1253  140-1261
LEVORF  140-1263  140-1264  140-1265
LEVORG  140-1260  140-1266
LEVORX  140-1260  140-1261  140-1263
LEVOWA  137-1213  137-1214
LEVOWE  137-1213  137-1215  137-1225
LEVOWR  137-1210  137-1212  143-1332  151-1530  176-2536  199-3476  203-3542
LEVOWS  137-1217  137-1220  137-1221  137-1222  137-1226
LEVOWW  137-1207  143-1336  143-1340
LEVOWX  137-1210  137-1211
LEV1RA  141-1303  142-1323
LEV1RB  141-1302  141-1304
LEV1RC  142-1315  142-1317
LEV1RD  141-1273  149-1434  149-1443  149-1447  150-1502  151-1536  200-3502
LEV1RE  142-1321
LEV1RF  141-1301  141-1304  142-1316  142-1320  142-1325  142-1330
LEV2.1  144-1341  150-1507  226-7651  226-7655  226-7667  227-7727  227-7735
LEV2.2  145-1343  148-1402  148-1413  148-1432  149-1442  149-1470
LEV2.E  144-1342  145-1345
LEV2.T  146-1347  148-1377  150-1521  199-3475
LEV2TA  146-1352  146-1357
LF.CON   58-0     154-1640
LF.SNR   58-0
LF.SUC   58-0
LFAIL    50-0
LGADR    65-0
LGCKSV   51-0      51-0
```



```
LM       48-0
LN       51-0      51-0
LNERR    70-0     142-1320  149-1436
LNPAG    15-0
LO.OFF   90-246    90-246    90-247
LO1     185-3027
LO1A    184-3024  184-3025  185-3030
LO2     185-3033
LO3     185-3037
LO4     185-3043
LOBYT    45-0      45-0      45-0      46-0      60-0      93-271    94-301   123-775   126-1052  127-1053  128-1054  129-1057  129-1060  132-1160
        133-1163  134-1164  135-1165  141-1275  141-1277  141-1303  141-1310  142-1323  146-1350  146-1356  148-1432  156-1730  156-1731  171-2453
        171-2454  173-2504  173-2506  176-2545  187-3141  189-3214  192-3303  211-3600  212-3613  217-3626  221-3772  221-4001  221-4006  226-7657
        227-7731  228-7737
LOG.LN   52-0      52-0
LOGCOD   45-0
LOGEND   52-0      52-0
LOGLEN   49-0      49-0
LOGPKT   52-0      52-0
LONIB    45-0     162-2113
LOPAR    90-246   224-7403
LSB      12-0      88-170    88-201   111-465   121-714   122-740   132-1153  142-1321  149-1451  149-1452  154-1634  155-1701  157-1747  164-2173
        182-2771
LT400    73-0
LV1SV1   50-0      50-0     141-1273  142-1327  142-1330
LV1SV2   50-0      50-0     141-1276  142-1326
LV2CNT   60-0     155-1676  157-1736  158-1774
LW       44-0
M        48-0
MAP.CH   63-0      63-0
MAP.MO   63-0      63-0
MAP.M1   63-0      64-0
MAP.ND   64-0      64-0
MAP.NX   63-0      63-0
MAP.OF   64-0      64-0
MAP.RD   64-0      64-0
MAP.S1   64-0      64-0
MAP.ST   64-0      64-0
MAP.UR   64-0      64-0
MAP.V0   64-0      64-0
MAP.V1   64-0      64-0
MAP.V2   64-0      64-0
MAP.V3   64-0      64-0
MAPENB   45-0
MAPFLG   51-0      51-0
MAPMSC   45-0
MAPMSH   45-0
MAPMSK   45-0
MAPSAV   51-0      51-0
MAPVAL   45-0
MAXSST  124-1022  125-1027
MAXTIM   93-266    93-270    93-271    94-300    94-301
MCP.IO   52-0
MCP.LN   52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0
```

```
          52-0      52-0      52-0      52-0      52-0      52-0      52-0
MCP.RD    52-0
MD.ALL    58-0
MD.CMP    58-0
MD.CSE    58-0
MD.ERR    58-0
MD.EXC    59-0
MD.EXP    58-0
MD.FEU    58-0
MD.FKC    59-0
MD.IMF    59-0     154-1615
MD.NOV    58-0
MD.NXU    58-0
MD.PRI    59-0     130-1101
MD.RIP    58-0
MD.SAV    58-0
MD.SCH    58-0
MD.SCL    58-0
MD.SEC    58-0
MD.SEQ    58-0
MD.SER    58-0
MD.SHD    58-0
MD.SPD    58-0     148-1374  148-1417
MD.SSH    58-0
MD.SWP    58-0
MD.VOL    58-0
MD.WBN    58-0
MD.WBV    58-0
MD512     45-0
MEMDST  220-3720  220-3742
MEMSZ     45-0      65-0
MERR      12-0
MINSST  124-1022  125-1023
MINTIM    93-266    93-266    94-300    94-301
MINUTE    45-0
MODE04  217-3636  219-3706
MODE4   219-3706  219-3711
MODE6   217-3637  219-3676
MODNUM    46-0      46-0      46-0
MP        73-0
MRQUE     49-0      49-0
MSB       13-0      88-213    91-257   136-1204  142-1320  146-1355  150-1501  151-1534  155-1676  156-1724  163-2144  165-2254  166-2317  169-2402
         174-2515  176-2541  181-2737  182-2764  190-3247  193-3333  198-3465
MSCPLN    51-0      51-0
MWQUE     49-0      49-0
NACLO     12-0      12-0      13-0
NBCB      65-0      65-0      65-0
NBIBAD    12-0
NBUFR     65-0      65-0      65-0
NCDONE    13-0
NCPE      13-0      13-0      13-0
NCRY      12-0      88-176   140-1253  164-2171  169-2376  179-2626  181-2676  182-2767  185-3026  217-3630  220-3733
NCSR      14-0      99-345   121-720   121-736   122-747   125-1025  137-1211  156-1706  180-2642  180-2643  180-2645  180-2647  180-2655  180-2656
         180-2660  180-2662
```



```
NDCLK     14-0     121-730   140-1255  164-2152  164-2154  164-2156  164-2160  164-2162  164-2164  164-2166  164-2170  164-2172  164-2174  164-2176
         164-2177  164-2201  164-2203  164-2205  164-2207  177-2550  177-2552  177-2554  177-2555  177-2557  177-2560  177-2561  178-2570  178-2572
         178-2574  178-2576  181-2677  181-2701  181-2703  181-2705  181-2707  181-2711  181-2713  181-2715  181-2717  181-2721  181-2723  181-2725
         181-2727  181-2731  184-3015  185-3027  185-3031  185-3033  185-3035  185-3037  185-3041  185-3043  185-3045  185-3056  185-3057  185-3060
         185-3062  196-3412  196-3416  196-3420  196-3421  196-3423  196-3424  196-3426  196-3430  196-3432  196-3434  196-3436  196-3440  196-3442
         196-3444  196-3446
NDPF      14-0      87-163   113-501   148-1371  187-3107  187-3111  187-3114
NDSER     14-0     101-372   124-1020  130-1112
NDSR      14-0      99-346
NEG       13-0      86-152    88-167    88-174    89-237    93-274    95-312    98-330    98-341   108-433   108-436   130-1113  155-1657  169-2354
         182-3000  188-3166  188-3170
NENDCD    69-0     143-1340
NEQ       12-0
NETST1    13-0
NFTEST    13-0      13-0
NLATE     14-0
NLSB      13-0      88-166    88-175    88-207    88-210   121-710   130-1101  137-1214  137-1215  148-1374  155-1662  156-1705  156-1707  156-1733
         163-2125  163-2134  164-2220  166-2303  166-2310  166-2312  174-2514  179-2602  179-2614  180-2637  180-2663  181-2735  182-3011  218-3663
NMERR     13-0
NMSB      12-0      88-212   132-1160  136-1171  155-1674  157-1767  158-1774  160-2040  164-2213  165-2224  165-2242  165-2267  167-2334  171-2443
         178-2566  179-2624  181-2736  194-3363  196-3452  221-3776  221-4000
NNEG      12-0      83-74     88-173    88-177    88-200    98-342   108-431   217-3634
NODEID    42-0
NOPREG    18-0
NOVER     14-0     104-402
NPKTS     52-0
NPOLL     12-0      13-0      13-0
NPRDY     14-0     106-412   106-414   111-471
NRPE      14-0
NRPOK     12-0      13-0
NRRDY     14-0      14-0
NSCAN     12-0      13-0      13-0      13-0
NSCB      65-0
NSDI      65-0      65-0
NSECTR    14-0
NSEEKS    49-0      49-0     126-1036  159-2016
NSTOP     13-0
NTEST     13-0      83-103    83-104    85-142    85-144
NUMFLG  223-4020
NUPF      12-0
NWRC      14-0     104-403   111-472   140-1250  181-2673  184-3023
NZRO      12-0      78-23     78-27     81-52     82-55     82-61     83-65     83-71     83-76     83-102    83-110    84-114    84-132    84-133
          85-147    87-157    88-171    88-203    89-235    89-236    90-243    90-247    91-256    93-270    93-273    94-276    95-307    98-331
          98-334    98-340    99-347   105-404   106-424   107-427   108-435   113-503   121-700   121-702   121-717   121-723   122-741   122-751
         122-753   122-754   122-760   122-763   122-764   122-765   123-774   124-1013  125-1026  125-1032  126-1037  126-1042  130-1076  132-1144
         132-1146  132-1154  136-1200  136-1202  136-1205  137-1223  140-1265  141-1274  141-1301  141-1302  141-1304  141-1306  142-1316  142-1323
         142-1325  143-1335  146-1352  148-1404  148-1407  148-1410  149-1437  149-1453  150-1516  150-1517  150-1522  150-1523  151-1531  153-1545
         153-1551  153-1565  153-1576  153-1603  153-1604  153-1605  154-1624  154-1643  155-1656  155-1657  155-1663  155-1666  155-1671  156-1720
         156-1722  157-1740  157-1742  157-1745  157-1752  157-1755  157-1761  157-1762  158-1777  158-2000  158-2007  159-2020  159-2030  160-2057
         160-2063  160-2067  161-2073  161-2101  163-2136  163-2143  164-2215  165-2230  165-2233  165-2243  165-2246  165-2257  166-2275  166-2300
         166-2314  167-2324  169-2355  169-2405  169-2411  169-2421  171-2427  171-2435  171-2447  171-2450  174-2511  174-2516  175-2527  175-2532
         177-2551  177-2556  177-2563  178-2571  178-2575  179-2601  179-2616  179-2623  179-2625  181-2700  181-2720  182-2755  182-3007  184-3017
         186-3073  186-3074  187-3101  187-3105  187-3106  187-3115  187-3124  187-3130  187-3134  187-3135  187-3136  187-3137  188-3147  188-3156
         188-3172  189-3233  189-3236  190-3271  193-3324  194-3361  195-3373  196-3414  196-3433  199-3475  200-3505  201-3520  201-3523  203-3543
```

```
        203-3546 206-3560 206-3563 206-3564 217-3635 217-3637 218-3647 218-3656 218-3662 219-3706 219-3715 220-3721 220-3722 220-3727
        220-3730 220-3735 220-3746 221-3754 221-3755 221-3756 221-3757 221-3765 221-4005 221-4011 221-4012 221-4016 226-7640
OBCLR   220-3736
OBSET   220-3731
OBTST   220-3724 220-3726
OCLK    105-405  114-506  114-507  114-510  114-511  114-513
OCLK00  104-377  114-513
OCLK1   100-380  104-401  114-510
OCLK48   98-353  100-356  114-507
OCLKRS  100-355  114-506  116-515
OCOMP   220-3725
ODDP     22-0     82-53    82-60    83-67    83-110   84-114   84-134   93-274   96-317  228-7770 228-7776
ODTAF    45-0    221-4003 221-4003
ODTCA    45-0
ODTCY    45-0    221-4003
OFFMSK   45-0
OFFSET   45-0
OFFTRK   60-0
OJCC    221-3756 221-3764
OLOAD   220-3723
ONLINE   69-0    149-1470
ONLREC   45-0
ONLY    114-514  114-514  115-514  115-514  118-535  118-535
OP.ABO   53-0
OP.ACC   53-0    126-1040
OP.ACP   58-0    154-1626
OP.ATT   53-0
OP.AVA   58-0    155-1663
OP.AVL   53-0    126-1046
OP.CCD   53-0
OP.CMP   53-0    126-1040
OP.DAP   53-0    126-1047 156-1717
OP.DUP   58-0    154-1624 154-1627
OP.END   58-0
OP.ERS   53-0    126-1040
OP.FLU   53-0
OP.GCS   53-0
OP.GST   53-0    126-1051
OP.GUS   53-0
OP.MRD   53-0
OP.MWR   53-0
OP.ONL   53-0    126-1045
OP.RD    53-0    126-1040
OP.RPL   53-0    126-1040
OP.SCC   53-0
OP.SUC   53-0    126-1050
OP.WR    53-0    126-1040
OPCOD    46-0
OPCODE   49-0     49-0
OPDCD   217-3645 220-3716
OPDCD1  218-3650 218-3651 220-3717
OPDCD2  220-3720 220-3743
OPM      22-0     83-101   84-135
OPNFLG  223-4020
```



```
OPX01   220-3730
OPX10   220-3722 220-3724
OPX11   220-3727 220-3735
OPXX0   220-3722
OPXX1   220-3721 220-3727
OSUBT   220-3735 220-3737
OTADD   220-3730 220-3732
OVER     14-0    111-466  116-520
OW       44-0
OWN      35-0
OXFC    221-3765 221-3772
OXFC0   221-3775 221-4003
OXFC17  217-3633 221-3776
P        52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0
         52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     53-0     53-0     53-0     54-0     54-0     54-0
         54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0
         54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     55-0     55-0     55-0     55-0     55-0
         55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0
         55-0     55-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0
         57-0     57-0     57-0     57-0
P.BCN0  54-0     54-0
P.BCN1  54-0
P.BUF0  54-0     54-0     54-0
P.BUF1  54-0     54-0     54-0
P.BUF2  54-0     54-0    189-3241
P.BUF3  54-0     54-0    189-3242
P.BUF4  54-0     54-0     54-0
P.BUF5  54-0     54-0
P.BUFH  54-0
P.BUFL  54-0
P.CMS0  55-0     55-0
P.CMS1  55-0
P.CNT0  55-0     55-0
P.CNT1  55-0     55-0
P.CNT2  55-0     55-0
P.CNT3  55-0
P.CNTF  54-0
P.CONS  54-0
P.CRF0  53-0     53-0     62-0
P.CRF1  53-0     62-0
P.CSVR  55-0
P.CTM0  55-0
P.CYL   55-0
P.CYLS  54-0
P.ELG0  54-0     54-0
P.ELG1  54-0
P.FBB0  55-0     55-0
P.FBB1  55-0
P.FLGS  55-0
P.GRP   55-0
P.HST0  54-0     54-0
P.HST1  54-0
P.HTM0  54-0
P.LBN0  54-0     54-0
```

LSCS FORM=QUAD

LSCS FORM=QUAD

CROSS REFERENCE TABLE (CREF V01-08)

```
        137-1223 137-1224 137-1225 140-1244 140-1246 140-1253 140-1255 140-1260 140-1267 140-1272 141-1301 142-1312 142-1313 142-1320
        142-1325 142-1331 144-1342 145-1346 146-1357 148-1372 148-1373 148-1400 148-1406 148-1407 148-1417 149-1436 149-1437 149-1441
        149-1461 150-1501 150-1503 150-1507 150-1511 150-1512 150-1513 150-1515 150-1516 150-1523 151-1531 151-1532 151-1533 151-1534
        151-1540 153-1564 153-1567 153-1570 153-1571 153-1572 153-1573 153-1601 153-1603 153-1607 154-1634 154-1635 154-1637 154-1642
        154-1643 154-1645 154-1647 154-1650 155-1657 155-1671 155-1672 155-1673 155-1676 156-1704 156-1706 156-1710 156-1712 156-1722
        156-1732 157-1737 157-1741 157-1755 157-1760 158-1776 158-1777 158-2005 158-2010 159-2012 159-2014 159-2015 159-2024 159-2031
        159-2033 160-2036 160-2041 160-2050 160-2053 160-2063 160-2066 160-2070 161-2073 162-2111 162-2113 162-2114 162-2122 163-2126
        163-2125 163-2130 163-2143 163-2146 164-2150 164-2151 164-2152 164-2154 164-2154 164-2156 164-2156 164-2164 164-2173 164-2174
        164-2177 164-2201 164-2203 164-2203 164-2207 164-2211 164-2211 165-2231 165-2241 165-2242 165-2245 165-2246 165-2247 165-2250
        165-2252 165-2261 165-2263 165-2270 166-2277 166-2301 166-2303 166-2304 166-2307 166-2310 167-2324 169-2353 169-2355 169-2356
        169-2357 169-2375 169-2404 169-2423 171-2427 171-2434 171-2437 171-2443 171-2446 171-2447 171-2453 171-2454 172-2472 174-2507
        174-2510 174-2511 174-2512 174-2514 174-2515 174-2517 175-2521 175-2523 175-2526 176-2547 177-2550 177-2553 177-2555 177-2561
        177-2562 178-2565 178-2570 179-2601 179-2602 179-2606 179-2623 180-2642 180-2643 180-2645 180-2646 180-2655 180-2656 180-2660
        180-2661 181-2673 181-2705 181-2707 181-2711 181-2713 181-2715 181-2723 181-2725 181-2727 181-2731 181-2736 182-2746 182-2747
        182-2750 182-2751 182-2752 182-2756 182-2770 182-2771 184-3020 185-3051 185-3051 185-3060 185-3062 185-3062 186-3065 186-3067
        186-3070 186-3073 186-3073 186-3074 187-3078 187-3077 187-3101 187-3102 187-3107 187-3111 187-3112 187-3114 187-3120 187-3132
        187-3133 187-3142 187-3146 188-3151 188-3152 188-3157 188-3172 188-3172 188-3174 188-3174 188-3176 189-3222 189-3232 189-3232
        190-3243 190-3245 190-3251 190-3251 190-3252 190-3253 190-3255 190-3256 190-3261 190-3262 190-3267 190-3267 190-3270 192-3310
        192-3311 193-3323 193-3327 193-3330 193-3333 193-3334 193-3340 193-3341 194-3343 194-3360 195-3403 196-3405 196-3424 196-3432
        196-3436 196-3436 196-3440 196-3440 196-3442 196-3442 196-3444 196-3444 196-3446 196-3450 196-3450 196-3453 197-3460 197-3461
        197-3462 200-3503 201-3521 201-3524 201-3526 202-3531 202-3532 202-3533 202-3534 202-3535 202-3536 202-3537 203-3546 205-3554
        205-3555 206-3560 210-3574 210-3575 210-3577 211-3600 211-3601 212-3603 213-3614 221-4000 226-7653 226-7661 226-7664 227-7724
        227-7726 227-7726 228-7756 228-7757 228-7760 228-7760 228-7761 228-7761 228-7762 228-7762 228-7763 228-7763 228-7764 228-7764
        228-7765 228-7767 228-7774 228-7774 228-7775
R12      17-0     17-0     89-226   89-227   93-270   93-271   93-273   94-303   101-373  104-377  104-377  106-411  106-411  109-443
        109-446  111-464  113-500  113-501  117-526  117-530  117-533  117-534  121-712  121-722  121-724  121-726  121-732  121-733
        122-740  122-742  122-747  122-750  122-752  122-753  125-1030 126-1040 126-1041 126-1042 126-1045 126-1046 126-1047 126-1050
        126-1051 129-1056 130-1072 130-1104 130-1112 131-1120 131-1130 131-1137 131-1140 132-1146 132-1153 136-1200 136-1201 137-1226
        137-1226 137-1227 137-1227 137-1230 137-1230 137-1231 137-1231 137-1232 137-1232 137-1233 137-1233 137-1234 137-1234 137-1235
        137-1235 137-1236 137-1236 137-1237 137-1237 138-1240 138-1240 138-1241 138-1241 138-1242 138-1242 138-1243 140-1246 140-1250
        140-1251 140-1252 140-1254 140-1256 140-1261 140-1262 140-1263 140-1264 141-1274 141-1277 141-1300 141-1304 141-1307 141-1310
        141-1311 142-1314 142-1315 142-1322 143-1332 143-1333 143-1336 143-1337 143-1340 144-1341 145-1343 145-1344 146-1350 146-1351
        146-1355 146-1356 147-1362 147-1363 147-1364 147-1365 147-1366 148-1371 148-1374 148-1401 148-1403 148-1413 149-1440 149-1452
        149-1454 149-1455 149-1457 149-1470 150-1471 150-1472 150-1474 150-1475 150-1505 151-1527 151-1530 152-1544 153-1547 153-1566
        153-1567 153-1576 153-1602 154-1621 154-1621 154-1622 154-1623 154-1624 154-1626 154-1627 155-1656 155-1663 155-1677 156-1711
        156-1716 156-1717 156-1720 156-1724 156-1725 156-1730 156-1731 156-1733 157-1734 157-1736 157-1752 157-1754 157-1765 157-1766
        157-1770 158-2000 159-2027 160-2041 160-2042 160-2054 160-2055 160-2056 160-2057 160-2060 160-2065 160-2067 160-2070 161-2077
        161-2100 161-2103 161-2106 161-2106 162-2115 162-2116 163-2137 163-2140 164-2162 164-2165 164-2171 164-2172 164-2173 164-2205
        164-2205 165-2236 165-2251 165-2270 165-2271 166-2275 166-2304 166-2310 166-2315 166-2316 166-2317 166-2321 166-2322 169-2365
        169-2366 169-2367 169-2370 169-2374 169-2377 169-2402 169-2420 169-2421 171-2444 172-2470 173-2500 173-2504 173-2506 174-2513
        174-2516 175-2531 175-2533 176-2536 176-2537 176-2540 176-2541 176-2542 176-2543 176-2545 177-2550 177-2557 177-2560 177-2560
        178-2570 178-2576 179-2617 179-2620 179-2631 181-2714 181-2716 181-2717 181-2720 181-2721 181-2734 181-2735 182-2745 182-2747
        182-2753 182-2755 182-2773 182-2775 184-3015 184-3016 184-3017 184-3022 184-3023 185-3030 185-3030 185-3032 185-3034 185-3034
        185-3036 185-3040 185-3040 185-3042 185-3044 185-3044 185-3046 185-3047 185-3052 185-3052 185-3057 185-3063 187-3104 187-3145
        188-3154 188-3163 188-3166 189-3217 189-3223 189-3224 189-3231 189-3241 189-3242 190-3261 190-3262 190-3270 192-3277 192-3300
        192-3303 192-3305 192-3314 192-3315 193-3332 194-3347 194-3352 194-3355 194-3363 194-3364 194-3366 195-3404 196-3415 196-3416
        196-3421 196-3434 199-3476 201-3511 201-3512 201-3513 201-3514 201-3515 203-3541 203-3542 203-3547 206-3557 210-3573 212-3604
        212-3612 212-3613 213-3615 226-7651 226-7652 226-7655 226-7656 226-7662 226-7667 226-7670 226-7673 227-7705 227-7707 227-7716
        227-7727 227-7730 227-7735 228-7737 228-7740
R13      17-0     78-27    78-30    89-227   89-230   93-266   93-273   94-300   94-301   104-375  105-405  106-410  106-423  107-425
        109-442  109-447  111-455  111-456  111-457  111-460  111-461  111-462  111-463  111-471  114-506  114-507  114-510  114-513
        117-532  122-754  123-774  123-776  124-1007 124-1010 124-1011 125-1023 125-1025 125-1027 125-1031 130-1111 131-1122 131-1123
        131-1132 131-1141 136-1174 136-1202 136-1206 142-1321 142-1321 143-1333 143-1334 143-1335 143-1337 144-1341 145-1343 146-1351
        147-1367 148-1414 148-1415 148-1421 148-1422 149-1433 149-1434 149-1443 149-1446 149-1451 149-1463 150-1504 153-1547 153-1550
```


CROSS REFERENCE TABLE (CREF V01-08)

```
        154-1614 154-1615 154-1616 154-1617 154-1630 154-1640 154-1645 155-1654 155-1655 155-1656 155-1702 155-1703 156-1705 156-1713
        156-1714 156-1727 157-1747 157-1750 157-1751 157-1753 157-1756 157-1762 157-1763 157-1764 158-1774 158-2002 158-2003 158-2006
        159-2012 159-2014 159-2022 160-2071 161-2075 161-2076 161-2101 161-2101 161-2102 164-2153 164-2155 164-2210 164-2215 165-2233
        165-2243 165-2257 166-2303 169-2360 169-2361 169-2375 169-2376 169-2414 169-2416 171-2436 171-2441 173-2477 177-2554 177-2556
        177-2563 178-2573 181-2722 182-2742 182-2764 182-2767 182-2776 184-3024 184-3025 185-3027 185-3033 185-3037 185-3043 187-3122
        187-3124 188-3150 188-3152 188-3154 188-3161 188-3162 188-3163 188-3165 188-3166 188-3167 188-3170 188-3170 188-3177 188-3177
        189-3222 190-3263 190-3263 190-3264 190-3264 192-3310 192-3313 192-3315 194-3345 194-3356 194-3361 196-3426 203-3543 203-3545
        212-3605 213-3616 226-7660 226-7663 226-7666 226-7672 227-7675 227-7676 227-7700 227-7703 227-7704 227-7706 227-7713 227-7714
        227-7715 227-7717 227-7722 227-7725 227-7733
R14      17-0     89-230   89-231   99-346   100-361  100-362  100-363  100-364  101-367  101-370  101-371  104-400  104-401  104-403
        119-536  121-736  122-743  124-1015 124-1016 124-1017 124-1020 124-1021 130-1076 130-1101 130-1103 136-1177 141-1302 141-1302
        142-1317 149-1447 149-1453 150-1502 151-1536 153-1552 153-1554 153-1555 153-1556 153-1560 153-1561 153-1605 153-1606 154-1612
        154-1646 155-1655 156-1727 157-1745 157-1761 159-2017 159-2020 160-2052 160-2062 160-2062 161-2105 161-2105 163-2133 164-2166
        164-2213 165-2224 165-2227 165-2256 166-2273 166-2274 167-2326 167-2332 167-2333 167-2337 168-2346 168-2350 169-2366 169-2374
        169-2410 171-2437 171-2440 171-2443 171-2450 171-2451 171-2452 175-2520 175-2524 175-2527 175-2532 175-2534 177-2555 179-2613
        181-2724 181-2730 182-2745 182-2751 182-2755 182-2775 182-3000 182-3000 182-3002 182-3004 185-3031 185-3035 185-3041 185-3045
        186-3064 187-3103 187-3105 187-3106 187-3110 187-3113 187-3122 187-3124 187-3126 187-3130 188-3156 189-3210 189-3224 190-3250
        190-3254 190-3256 190-3266 190-3266 190-3271 190-3271 192-3277 194-3344 196-3430 201-3516 201-3516 201-3520 201-3520 201-3523
        201-3523 212-3606 213-3617 221-4003 221-4003 226-7674 227-7736 228-7741
R15      17-0     17-0     89-231   89-232   99-345   99-346   99-352   99-353   200-3477 200-3504 201-3517 212-3607 213-3620 221-4003
R16      17-0     89-232   89-233   99-350   100-355  100-356  100-357  100-365  100-366  101-372  106-414  107-427  108-430  108-432
        108-433  109-441  113-505  119-536  119-537  121-715  121-716  121-720  126-1034 130-1100 130-1106 131-1125 131-1126 131-1133
        131-1134 131-1135 132-1144 132-1160 132-1162 132-1162 141-1274 141-1275 141-1304 141-1305 141-1306 142-1323 142-1324 146-1347
        146-1347 146-1352 146-1354 146-1354 147-1361 148-1375 148-1376 148-1402 148-1416 149-1435 149-1440 149-1444 149-1456 149-1462
        150-1517 152-1542 153-1557 153-1575 153-1604 155-1662 155-1670 157-1740 157-1742 157-1743 158-2001 159-2026 159-2030 159-2031
        161-2104 161-2107 163-2127 163-2127 163-2141 163-2147 164-2150 164-2153 164-2221 165-2241 165-2244 167-2323 167-2325 167-2327
        167-2330 167-2331 167-2335 167-2336 167-2340 167-2341 167-2342 168-2346 168-2351 168-2351 169-2353 169-2400 169-2405 169-2415
        171-2427 172-2466 173-2501 175-2522 175-2525 175-2530 178-2574 179-2605 179-2605 180-2650 180-2652 180-2657 180-2660 181-2674
        181-2675 181-2677 181-2706 181-2733 182-2760 182-2762 182-3001 182-3006 182-3011 184-3020 186-3064 187-3103 187-3126 187-3130
        187-3132 187-3134 187-3135 187-3136 187-3141 187-3143 187-3144 188-3147 188-3150 188-3151 188-3165 188-3205 188-3206 188-3207
        189-3214 189-3215 189-3216 189-3220 189-3221 189-3225 190-3244 190-3246 190-3247 190-3260 190-3265 192-3274 192-3276 192-3320
        192-3322 194-3342 195-3367 195-3370 196-3437 196-3441 198-3466 199-3471 199-3475 201-3522 212-3610 213-3621 226-7657 227-7702
        227-7724 228-7745 228-7746
R17      37-0     76-0     76-2     78-22    78-23    78-24    78-25    78-26    81-52    89-223   89-224   106-422  108-432  108-433
        109-444  111-472  111-473  111-474  112-475  116-515  121-703  121-704  121-705  121-707  121-725  121-730  121-731  121-731
        124-1005 124-1011 130-1064 130-1070 130-1102 131-1121 131-1124 141-1303 141-1307 141-1311 142-1316 145-1344 146-1353 146-1353
        147-1360 147-1362 147-1370 148-1371 148-1374 148-1377 148-1402 148-1404 148-1410 148-1420 148-1425 148-1427 148-1430 148-1432
        149-1442 150-1521 152-1541 153-1546 153-1551 153-1553 153-1557 153-1564 153-1565 154-1652 155-1665 155-1666 159-2027 163-2134
        163-2136 163-2144 164-2155 164-2157 164-2175 164-2200 164-2202 164-2204 164-2212 165-2223 165-2230 165-2234 165-2255 165-2262
        165-2264 166-2272 166-2311 166-2314 167-2343 168-2345 168-2347 169-2352 169-2362 169-2403 169-2406 169-2417 169-2422 171-2430
        171-2462 171-2464 172-2467 172-2473 172-2474 173-2476 176-2544 176-2546 179-2607 179-2611 179-2614 179-2616 179-2621 179-2626
        179-2632 180-2636 180-2640 180-2643 180-2645 180-2651 180-2664 180-2665 180-2666 180-2667 180-2671 181-2704 181-2706 181-2710
        181-2712 181-2716 181-2737 182-2743 182-2754 182-2761 182-2763 182-2765 182-2774 182-3013 185-3053 185-3054 186-3071 187-3075
        187-3117 187-3131 188-3153 188-3160 188-3164 188-3202 189-3226 189-3227 189-3230 192-3273 192-3301 192-3306 193-3325 193-3326
        193-3331 193-3337 194-3346 194-3351 194-3362 195-3372 195-3374 195-3374 195-3375 195-3375 195-3376 195-3376 195-3402 196-3412
        196-3413 196-3414 196-3420 196-3422 196-3422 196-3425 196-3425 196-3427 196-3427 196-3433 196-3445 196-3452 201-3525 210-3571
        212-3611 213-3622 221-3756 221-4013 221-4016 223-4020 226-7643 226-7644 226-7645 226-7646 226-7647 226-7650 226-7665 226-7671
        227-7716 227-7723 227-7731 227-7732
R2       86-151   86-153   86-154   86-155   89-216   89-217   91-253   91-254   91-255   95-313   95-314   98-326   98-335   98-336
        212-3602
R3       17-0     76-5     89-217   89-220
R4       17-0     17-0     89-220   89-221
R5       17-0     89-221   89-222
R6       17-0     89-222   89-223   98-322   98-334   98-341
```

CROSS REFERENCE TABLE (CREF V01-08)

```
R7       17-0      89-233    89-234
RAMPE    22-0      83-101    84-135    91-261    96-320
RAMPET   95-305    96-315
RAMTST   83-75     98-322
RAMZAP   95-304    95-311    95-314    98-323
RBCAI    15-0
RBN2ND   46-0      46-0      46-0      46-0      46-0      130-1101  187-3136
RBNCLA   189-3222  189-3225
RBNCLB   189-3222  189-3226
RBNCLC   189-3211
RBNCOD   46-0      165-2256  169-2370  187-3146  189-3231
RBNMSK   46-0      169-2365  187-3145  188-3162  189-3221
RBNPRI   187-3110  189-3210  192-3307
RBNPRM   46-0      46-0      46-0      130-1076  187-3106  187-3135
RBNSTR   46-0      189-3217  190-3255
RBNXOR   46-0      46-0      46-0      46-0      187-3110
RBPA     223-4020
RCC      17-0      217-3643  217-3644  218-3654  218-3657  218-3657  218-3660  218-3662  218-3663  218-3663  218-3664  218-3666  218-3666  218-3667
         218-3667  218-3670  218-3670  218-3671  218-3671  218-3672  218-3673  218-3674  218-3674  218-3675  218-3675  219-3700  219-3701  220-3723
         220-3724  220-3725  220-3726  220-3730  220-3731  220-3732  220-3733  220-3735  220-3736  220-3737  220-3740  220-3743  220-3747  221-4003
         221-4012  221-4015
RCI      40-0
RCLR     73-0      76-7      100-361   101-373   116-523   116-524   124-1021  131-1115  140-1272
RCMD     16-0      104-403   140-1245  164-2150  177-2563  178-2570  184-3016
RCSRWO   47-0      47-0
RCSRW1   47-0      47-0      47-0      47-0
RCT.EN   46-0
RCT.MT   46-0
RCT.PR   46-0
RCT.SE   46-0
RCTBLK   51-0      51-0      192-3316
RCTOFF   50-0      51-0      192-3312
RCY      17-0      217-3630  218-3660  218-3661  218-3662  218-3665  220-3733  220-3734  220-3740  220-3741  221-3764  221-4003  221-4010
RD       24-0      180-2642  180-2655  188-3200  188-3201
RDCLK    109-451   110-453   113-500   137-1226  137-1227  137-1230  137-1231  137-1232  137-1233  137-1234  137-1235  137-1236  137-1237  138-1240
         138-1241  138-1242
RDCLKO   113-476   113-500   115-514
RDFST    15-0
RDNXT    15-0
RDPF     16-0      76-5      87-161    87-164    105-404   108-436   123-1003  126-1037  136-1174  148-1376  163-2124  179-2601  188-3157  195-3367
         198-3463  227-7677
READ     40-0
READCD   69-0      69-0      69-0      192-3305  193-3333
RECALB   69-0      150-1475  226-7662
RECC     16-0      116-517   117-532   137-1214  164-2207  165-2231  171-2427  182-2766  196-3446  196-3454
RECCT    16-0      106-422   110-452   116-515   164-2201  181-2725  196-3440
RELES    82-53     83-70     83-100    84-115    84-123    85-136    85-143    85-144    85-145    85-147
REPSTA   51-0      51-0      130-1067  131-1116  131-1131  177-2564
RETCNT   60-0      123-774   123-774
RETURN   221-3760
REVEND   49-0      49-0      49-0
REVSTR   48-0      48-0
RGDATH   51-0
RGDATL   51-0
```


CROSS REFERENCE TABLE (CREF V01-08)

```
RIB      17-0      217-3633  217-3634  217-3635  217-3636  217-3637  217-3641  218-3647  218-3652  218-3655  218-3656  219-3706  219-3713  219-3714
         220-3717  220-3720  220-3721  220-3722  220-3727  220-3744  220-3745  221-3753  221-3754  221-3755  221-3757  221-3762  221-3774  221-3775
         221-3776  221-3777  221-4000  221-4001  221-4002  221-4011
RLL      17-0      83-113    84-130    121-700   121-701   121-702   130-1063  132-1155  153-1562  154-1644  159-2021  160-2061  162-2120  163-2126
         164-2170  164-2170  164-2174  164-2214  164-2217  164-2220  164-2222  165-2232  166-2305  166-2312  166-2320  166-2320  169-2354  169-2407
         169-2407  171-2426  179-2604  179-2624  179-2625  179-2625  180-2634  180-2640  180-2662  180-2664  180-2670  180-2670  181-2741  182-2766
         182-2766  182-2772  182-2777  182-3010  182-3012  182-3012  187-3100  187-3140  188-3155  189-3213  189-3235  197-3456  206-3561  206-3562
         206-3563  206-3565  208-3567  208-3567  208-3570  209-3572  209-3572  217-3632  226-7640  226-7642  228-7766
RNGBSH   49-0      49-0
RNGBSL   49-0      49-0
ROL      218-3657
ROR      218-3656  218-3662
ROTATE   218-3647  218-3652
ROTR1    218-3674  218-3675
ROTR3    218-3666  218-3667  218-3670  218-3671  218-3674
RPC      17-0      217-3626  217-3627  217-3631  217-3631  217-3645  217-3645  217-3646  218-3653  218-3653  219-3677  219-3677  219-3705  220-3742
         220-3742  221-3752  221-3752  221-3756  221-3763  221-3763  221-3765  221-3765  221-3771  221-4005  221-4013  221-4014  221-4014  221-4016
         221-4017  221-4017
RPE      14-0
RPOK     12-0      13-0
RRDY     14-0      14-0      116-522   117-525
RRM      16-0      106-424   116-514   140-1267  182-2775
RRSGEN   16-0      116-521   182-3003
RSE      16-0      87-163    116-516   121-727   121-731   130-1074  137-1224  138-1243  140-1260  140-1265  163-2141  163-2142  164-2211  171-2424
         177-2561  179-2622  180-2633  181-2736  184-3015  185-3050  196-3412  196-3450  197-3455
RSPCOF   47-0      47-0
RSPLEN   47-0      47-0      47-0      47-0
RSPO     48-0
RSPPOF   47-0      47-0
RSPPTR   47-0      47-0
RT1      17-0      217-3637  217-3640  217-3642  217-3642  217-3643  217-3646  218-3650  218-3651  219-3676  219-3701  219-3702  219-3702  219-3703
         219-3704  219-3705  219-3706  219-3707  219-3710  219-3710  219-3711  219-3712  219-3715  220-3716  220-3723  220-3725  220-3726  220-3731
         220-3732  220-3736  220-3737  220-3746  220-3750  220-3751  221-3760  221-3761  221-3762  221-3767  221-3770  221-3772  221-3773  221-4004
         221-4005  221-4006  221-4007
RTCS     18-0      18-0      18-0      99-350    100-355   100-356   100-357   104-376   124-1006  124-1012  124-1013  124-1014  125-1022  125-1024
         140-1247  140-1253  140-1257  140-1270  156-1720  163-2132  163-2141  164-2207  179-2612  180-2641  180-2642  180-2643  180-2645  180-2654
         180-2655  180-2656  180-2660  181-2732  188-3200  188-3201  196-3410  196-3446
RTCSO5   24-0
RTCSO6   24-0
RTCS13   24-0
RTCS14   24-0
RTDS     18-0      18-0      18-0      99-345    121-712   121-736   122-747   124-1020  137-1213  137-1214  140-1272  151-1533  153-1605  155-1673
         156-1706  157-1761  161-2077  163-2137  163-2143  166-2316  167-2333  174-2507  174-2510  174-2511  174-2512  174-2513  174-2516  175-2523
         175-2524  175-2526  175-2527  175-2532  175-2534  176-2540  178-2565  179-2617  179-2623  181-2734  181-2735  184-3015  184-3016  184-3017
         196-3412  196-3413  196-3414  202-3531
RTDSO6   25-0
RTDSER   124-1015  124-1020
RTOEVE   43-0
RUNN     69-0      150-1505
RUPF     15-0      15-0      15-0
RVACTV   60-0
RVCBNH   49-0      49-0      192-3305
RVCBNL   48-0      49-0      192-3304  192-3305
RVCBUF   51-0      51-0      132-1154  132-1161  193-3336
```

```
RVCEBF   51-0      51-0     192-3275
RVCFLG   51-0      51-0     132-1150 132-1152 132-1157 192-3321
RVCIND   50-0      50-0
RVCRTY   50-0      50-0
RVCSAV   48-0      48-0     192-3300 194-3344
RVCSDI   51-0      51-0     132-1145 194-3353
RVCT     59-0     122-760   122-764  136-1204 165-2254 165-2267 169-2402 169-2402 171-2443 194-3356
RVCTA    49-0      49-0     192-3302
RVCVEC   51-0      51-0
RVECDP   51-0      51-0     132-1147 132-1156 192-3317
RVECUP   51-0      51-0     194-3354
RVWRIT   60-0
RWM      16-0
RWRDY    25-0      99-352   100-363  151-1533 155-1674 163-2144 166-2316 167-2334 174-2515 175-2520 176-2540 178-2566 179-2624 181-2736
S.ADRH   54-0
S.ADRL   54-0      54-0
S.CFL1  147-1367 227-7677 227-7721
S.CL11  159-2015 189-3217 190-3256
S.CLRB   96-316    96-317  130-1067 141-1275 148-1414 155-1701 160-2064 165-2265 169-2413 182-2742 187-3116 194-3353 194-3354 214-3623
S.CMOD  147-1360 227-7675 227-7714
S.CYLH   54-0      54-0
S.CYLL   54-0      54-0     192-3276 194-3342 194-3366
S.DCLR  148-1401 226-7657
S.DCN1  148-1403 227-7732
S.DCN2  148-1414 227-7724
S.DCN3  148-1411 155-1666
S.DCNE  148-1404 148-1410 148-1413 148-1420 148-1425
S.EREC  148-1426 227-7705
S.GRUP   54-0      54-0
S.GUCH  149-1441 226-7673 228-7740
S.INIT  149-1465 226-7654
S.LBNH   54-0     165-2261
S.LBNL   54-0     165-2263
S.LD11  121-737  123-771  123-772  123-1000 130-1064 130-1100 136-1172 136-1203 142-1327 148-1405 149-1460 150-1476 150-1500 150-1506
        150-1510 154-1633 155-1661 155-1667 155-1675 155-1700 157-1757 158-1775 159-2023 159-2032 160-2047 162-2112 165-2240 165-2244
        165-2253 165-2260 165-2266 166-2276 166-2306 169-2373 169-2401 169-2403 169-2422 171-2433 171-2445 171-2460 176-2546 179-2603
        186-3072 187-3075 187-3117 187-3131 188-3171 190-3254 193-3326 193-3331 193-3337 194-3357 201-3515 209-3571 212-3603
S.LD12  129-1055 132-1145 136-1173 142-1326 146-1347 150-1473 152-1543 153-1574 153-1577 154-1620 154-1621 155-1654 156-1715 156-1721
        156-1723 160-2037 160-2040 165-2235 169-2364 169-2372 169-2417 172-2467 172-2470 173-2502 176-2544 179-2630 182-2757 187-3144
        189-3211 189-3216 189-3234 192-3301 193-3325 194-3346 194-3351 194-3365 201-3510 206-3556 212-3604
S.LD13  123-767  126-1035 126-1036 130-1106 131-1131 132-1157 153-1546 154-1613 154-1614 155-1653 157-1744 157-1746 157-1773 161-2074
        161-2105 169-2352 180-2651 180-2666 185-3053 188-3160 188-3161 188-3164 195-3375 200-3500 212-3605
S.LD14  136-1170 136-1171 154-1610 156-1726 165-2255 165-2262 165-2264 166-2311 168-2345 168-2347 171-2442 171-2461 171-2463 182-2776
        185-3054 188-3153 190-3243 190-3245 195-3376 200-3501 201-3511 212-3606
S.LD15  201-3512 212-3607
S.LD16  122-755  130-1077 132-1152 132-1156 136-1175 153-1600 155-1660 159-2025 172-2465 173-2476 189-3220 190-3272 192-3273 192-3317
        192-3321 201-3513 212-3610
S.LD17  123-777  124-1004 132-1151 136-1176 148-1416 148-1426 148-1427 148-1431 168-2344 171-2432 171-2435 173-2475 182-2760 182-2762
        193-3336 195-3371 195-3377 198-3464 199-3474 201-3514 212-3611 221-3752
S.LDCC  212-3608 220-3742 220-3744
S.LDCY  212-3607
S.LDDB  131-1121 131-1127 165-2234 167-2343 169-2362 171-2430 171-2431 172-2473 186-3071 189-3227 195-3374 212-3607
S.LDPC  212-3604 221-3761 221-4007
S.LDR2   95-313  212-3602
```



```
S.LL12  126-1034 156-1716 187-3141 189-3214 195-3401 212-3612
S.LRSP  150-1475 150-1506 226-7662
S.ML12  212-3613
S.ONLN  149-1467 226-7665
S.OPFL   54-0
S.RCAL  150-1471 227-7712
S.ROTL  162-2115 182-2746 182-2752 210-3574 210-3575
S.RR17  142-1312 149-1441 210-3577
S.RUNN  150-1504 226-7671
S.SD11  121-703  153-1563 160-2052 211-3600
S.SECI   54-0     54-0    130-1106 192-3276 194-3342
S.SECS   54-0     54-0
S.SEEK  150-1510 227-7702
S.SND3  147-1366 148-1371 148-1375 148-1375
S.SRM   106-423  107-426  117-531
S.ST11  131-1116 136-1166 142-1331 150-1477 159-2034 160-2043 171-2455 186-3073 188-3172 189-3226 192-3312 213-3614 226-7641
S.ST12  148-1378 149-1445 154-1631 155-1656 157-1735 160-2041 162-2117 165-2237 172-2471 176-2547 181-2740 189-3212 192-3302 192-3306
        192-3316 200-3506 213-3615
S.ST13  148-1375 158-2004 161-2106 189-3237 189-3241 192-3304 213-3616
S.ST14  130-1103 156-1727 159-2020 165-2261 165-2263 171-2462 171-2464 182-3000 189-3240 189-3242 192-3305 213-3617
S.ST15  213-3620 221-4003
S.ST16  132-1147 141-1277 160-2044 172-2466 182-3013 190-3257 192-3275 192-3320 213-3621
S.ST17  142-1321 164-2213 169-2412 171-2425 195-3400 213-3622
S.STCC  213-3617
S.STDB  130-1061 160-2035 213-3620
S.STPC  213-3615 221-3773
S.SWAB  146-1350 146-1355 147-1364 148-1372 173-2504 192-3315 210-3575
S.TRAK   54-0     54-0
SAREG    41-0
SAVADR   51-0     51-0
SAVBUF   51-0     51-0
SAVCNT   51-0     51-0
SAVEDC   51-0     51-0
SAVR7    50-0     50-0
SAVUAR   51-0     51-0
SC.AOL   56-0
SC.BAD   56-0
SC.CNT   57-0
SC.DCL   56-0    167-2325
SC.DDE   57-0    149-1462 157-1740
SC.DIS   56-0
SC.DSY   56-0
SC.DUP   56-0
SC.EC1   56-0
SC.EC2   56-0
SC.EC3   56-0
SC.EC4   56-0
SC.EC5   56-0
SC.EC6   56-0
SC.EC7   56-0
SC.EC8   56-0
SC.ECC   56-0
SC.EDC   57-0
SC.FER   56-0
```

```
SC.HDR   56-0     169-2400 169-2405 169-2415
SC.IMR   57-0
SC.INV   56-0     157-1743
SC.IOP   56-0
SC.LVO   57-0     131-1133 167-2330
SC.N12   56-0
SC.NML   56-0
SC.NOM   57-0
SC.NVL   56-0
SC.ODB   57-0
SC.ODT   57-0
SC.OVR   57-0     167-2323
SC.PAR   57-0
SC.POS   56-0     167-2336
SC.RRD   57-0     167-2335
SC.RWR   56-0     131-1134 167-2331
SC.SCN   56-0
SC.SDI   56-0
SC.STO   56-0     157-1742
SC.UNK   56-0
SC.WPH   56-0
SC.WPS   56-0
SCAN     12-0      12-0      12-0      13-0
SCLR     73-0      76-0     100-355   101-372   116-522   124-1020  130-1114  140-1271  228-7774
SCMD     16-0     104-377   121-725   130-1073  137-1213  140-1266  163-2140  165-2233  179-2620  185-3052  196-3407
SD       18-0     104-377   104-403   106-411   111-473   112-475   117-532   121-731   137-1217  137-1226  137-1227  137-1230  137-1231  137-1232
        137-1233  137-1234  137-1235  137-1236  137-1237  138-1240  138-1241  138-1242  140-1251  140-1252  140-1254  140-1256  140-1261  140-1263
        164-2152  164-2154  164-2156  164-2160  164-2162  164-2164  164-2166  164-2170  164-2172  164-2174  164-2176  164-2177  164-2201  164-2203
        164-2205  177-2550  177-2552  177-2554  177-2555  177-2557  177-2560  178-2570  178-2572  178-2574  178-2576  181-2673  181-2674  181-2675
        181-2677  181-2701  181-2703  181-2705  181-2707  181-2711  181-2713  181-2715  181-2717  181-2721  181-2723  181-2725  181-2727  181-2731
        184-3024  184-3025  185-3027  185-3031  185-3033  185-3035  185-3037  185-3041  185-3043  185-3045  185-3056  185-3057  185-3060  185-3062
        196-3416  196-3420  196-3421  196-3423  196-3424  196-3426  196-3430  196-3432  196-3434  196-3436  196-3440  196-3442  196-3444
SD.RS    109-440   116-516
SD.RST   104-374   106-407   116-514
SDCLK    106-413   106-415   106-416   106-420   106-422   109-443   109-445   113-501   113-503
SDI.1     65-0      65-0     154-1634  155-1701  157-1747  211-3600  211-3601
SDI.12    63-0      63-0     187-3141  189-3214
SDI.1T    60-0      60-0     150-1517  154-1631  160-2043
SDI.2     65-0      65-0     154-1634  155-1701  157-1747
SDI.2T    62-0
SDI.3     65-0      65-0     154-1634  155-1701  157-1747
SDI.4     65-0      65-0     152-1541  154-1634  155-1701  157-1747  227-7731
SDI.AT    62-0      62-0      62-0      62-0      62-0      62-0      62-0      62-0     154-1634  155-1701  157-1747
SDI.BH    61-0      61-0
SDI.BL    61-0      61-0
SDI.CH    60-0      60-0     154-1651
SDI.CL    60-0      60-0
SDI.CP    59-0      60-0     188-3204
SDI.CW    62-0      63-0     136-1176  159-2034  188-3206
SDI.DB    60-0      60-0     154-1620  164-2213  168-2344  171-2432  173-2475  182-2764
SDI.DL    61-0     227-7736
SDI.DP    63-0      63-0     173-2502
SDI.EO    60-0      60-0     136-1170  155-1675  155-1701  156-1723  157-1735  157-1773  158-2004  160-2037  165-2265  171-2445
SDI.E1    60-0      60-0     123-767   148-1431
```



```
SDI.EC    61-0     150-1500 150-1506
SDI.ER    61-0      61-0      61-0
SDI.ES    60-0      61-0     123-1000  150-1473  155-1667  166-2306  171-2433
SDI.FC    63-0      63-0
SDI.GC    63-0      63-0     187-3143  189-3215
SDI.GO    63-0      63-0
SDI.GP    60-0      60-0     151-1526  176-2547
SDI.H1    63-0      63-0     149-1443  226-7674  228-7741
SDI.H2    63-0      63-0
SDI.I1    61-0      61-0
SDI.I2    61-0      61-0
SDI.I3    61-0      61-0
SDI.L1    63-0      63-0     169-2373
SDI.L2    63-0      63-0     169-2372
SDI.LL    63-0     226-7674  228-7741
SDI.LN    52-0
SDI.LT    61-0      61-0      61-0     162-2121
SDI.M1    63-0      63-0
SDI.M2    63-0      63-0
SDI.OE    60-0      60-0     169-2413  182-2742  182-2776
SDI.OM    60-0      60-0     126-1035  157-1744
SDI.OP    62-0
SDI.PH    63-0      63-0      63-0     132-1151  169-2412
SDI.PL    62-0      62-0
SDI.PQ    60-0      60-0     122-755   130-1077  136-1172  154-1613  154-1633  155-1700  156-1715  157-1746  165-2260  171-2460  189-3234  194-3365
SDI.RC    61-0
SDI.RH    61-0      61-0     171-2463  189-3240
SDI.RL    61-0      61-0     171-2461  189-3237
SDI.RO    60-0      60-0     165-2237  188-3171
SDI.RS    61-0      61-0
SDI.RT    63-0      63-0     159-2023  169-2364  187-3144  188-3161  189-3220
SDI.RV    63-0      63-0
SDI.S1    61-0      62-0     121-737   136-1173  149-1460  153-1577  155-1661  161-2074  227-7712
SDI.S2    62-0      62-0     136-1175  149-1457  153-1600  155-1660
SDI.S4    62-0     149-1457
SDI.S5    62-0
SDI.S6    62-0
SDI.S7    62-0
SDI.SD    63-0      63-0     160-2035
SDI.SL    59-0      59-0     154-1611  155-1653  159-2032  164-2161  165-2235  172-2470  181-2722  189-3211
SDI.SS    60-0      60-0
SDI.ST    59-0      59-0     121-711   132-1142  136-1203  148-1405  150-1510  153-1553  157-1757  158-1775  160-2047  160-2053  164-2167  165-2253
        165-2266  166-2276  169-2401  171-2442  179-2630  182-2767  194-3357  209-3571
SDI.SV    60-0      60-0     136-1166  136-1171  156-1726  160-2040  171-2455
SDI.SW    60-0      60-0     129-1055  160-2044
SDI.TG    63-0      63-0     188-3147  189-3216
SDI.TI    61-0      61-0      61-0
SDI.TM    60-0      60-0     156-1721  162-2117
SDI.TO    61-0      61-0      61-0      61-0     148-1411  148-1423  149-1434  162-2112
SDI.UB    60-0      60-0     123-771   123-777   148-1426  165-2240  171-2435  179-2603
SDI.UE    61-0      61-0
SDI.UF    63-0      63-0     148-1414  159-2025  160-2064
SDI.UG    60-0      60-0
SDI.UN    61-0      61-0     149-1455  152-1543  153-1546  153-1574  154-1610  160-2045  228-7742
```

```
SDI.V1   63-0      63-0
SDI.V2   63-0      63-0
SDI.XH   60-0      60-0
SDI.XL   60-0      60-0
SDI.XR   61-0
SDIB.L   64-0      64-0      65-0      65-0      65-0      65-0      121-710   153-1565  160-2063
SDIBEG   65-0      65-0      65-0
SDICL1   99-350    100-361   100-365
SDICL2   99-351
SDICLK   99-345    100-362   100-363   100-364   101-367   101-370   101-371   101-373
SDIRTY   69-0      157-1752  157-1754  157-1770
SDIS     46-0      121-704   124-1005  124-1010
SDITMO   69-0      226-7665
SDITO    46-0
SDITST   83-101    83-106    100-354
SDPF     16-0      87-162    109-443   130-1062  153-1545  153-1555  195-3373  198-3465  227-7721
SDRD     108-430   114-514
SDRRDY   116-522   117-530
SDTIMO   69-0
SDWRT    104-374
SEC      25-0      99-353    101-367   130-1066  130-1110  131-1124  175-2521  182-2756  185-3063  188-3176  195-3403  196-3452  205-3554
SECC     16-0      104-376   106-411   109-442   178-2572  180-2652  180-2667
SECCT    16-0      104-401   106-417   109-446   164-2152  181-2701  196-3416
SECSZ    46-0      65-0      131-1125
SECTR    14-0
SEEK     59-0      122-761   122-763   154-1616  155-1702  157-1763  166-2277
SEEKGS   150-1517  151-1526
SEEKL    69-0      150-1521
SEKPRV   48-0      48-0
SEKSAV   48-0
SEQ01    78-30     81-52
SEQERR   76-3      76-4      76-5      76-6      76-10     77-12     78-26     119-541   224-7400  226-7416
SEQTST   78-22     78-22
SEQUEN   46-0
SERIAL   22-0      91-254    91-257
SES      42-0
SETCRY   217-3630  218-3660  220-3734
SEX2SK   48-0
SF       73-0
SFERR    70-0      141-1301
SFT20    89-214    89-236
SGRPCD   69-0      151-1530  176-2536
SIMTST   1-0       83-103
SLAT     59-0      121-720   121-721   122-757   131-1140  150-1515  154-1630  154-1642  154-1643
SM       73-0
SNGLOP   217-3635  218-3647
SRM      16-0      104-402   117-531   140-1246  184-3017
SRSGEN   16-0      116-520   182-3002
SSE      16-0      104-375   106-410   109-441   121-726   137-1216  140-1250  164-2151  178-2567  181-2673  184-3020  185-3055  196-3415
SST      42-0
ST.ABO   55-0
ST.AVL   55-0
ST.CMD   55-0
ST.CMP   55-0
```



```
ST.CNT   56-0      167-2323
ST.DAT   55-0      169-2400  169-2405  169-2415
ST.DRV   56-0      129-1056  131-1135  149-1462  157-1740  157-1742  157-1743  167-2325  167-2327  167-2331  167-2335  167-2336
ST.HST   55-0
ST.MFE   55-0      129-1056
ST.MSK   55-0
ST.OFL   55-0
ST.SHF   55-0
ST.SUB   55-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0
         56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0      56-0
         56-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0      57-0
ST.SUC   55-0
ST.WPR   55-0
STARSP   69-0      226-7656  226-7670  227-7730
START    114-514   114-514   114-514   114-514   115-514   118-535   118-535
STCACH   65-0      65-0
STDALN   46-0
STEP     77-11     77-15
STEP1    35-0      38-0      79-33     80-40
STEP2    35-0
STEP3    35-0
STEP4    35-0
STEST    13-0
STOP     12-0
STOPCM   43-0
STOPEN   43-0
STORB    220-3747
STORCC   218-3663  218-3665  218-3673
STORE    219-3714  220-3744
STORT1   213-3621  219-3712  221-3762  221-3767
STRT.F   143-1332  144-1341  145-1343  146-1351
STRTCD   69-0      141-1300  143-1332
STS      42-0
SUPF     15-0      15-0      15-0
SUSP     59-0      122-761   123-776   131-1130  155-1702  157-1765  160-2071  162-2111  165-2251  166-2275  169-2377  171-2444  173-2500  194-3345
         194-3355  194-3364
SWAB     218-3662  218-3666
SWM      16-0      121-724   130-1072  137-1212  163-2133  163-2143  163-2147  179-2613  179-2623  182-2765  185-3051  196-3406
SY       48-0      48-0      48-0
SY0      48-0
SY8      48-0      48-0
SYNC     50-0      50-0      137-1215  177-2551  178-2571  184-3014  196-3411
SYNCH    46-0      117-534
SYNCL    46-0      117-533
T3       82-54     82-56     224-7407
T4       82-60     82-62     224-7410
T5       83-64     83-66     224-7411
T6       83-67     83-72     224-7412
T7       83-75     83-77     224-7413
T9       83-101    83-112    224-7415
TEMP     51-0      51-0
TEMP1    48-0      48-0
TEMP2    48-0      48-0
TEMP3    48-0      48-0
```

```
TEST     12-0      85-143
TESTBL  121-703   224-7404
THEN    114-514   114-514   115-514
TMOUT    93-271    94-276    94-303
TMR.BS   50-0      50-0
TMR.MC   50-0      50-0     150-1477
TOPOL    69-0     226-7651
TOPRSP   69-0     226-7652
TPO10   106-411   114-514
TPO11   106-417   114-514   114-514
TPO12   109-442   115-514
TPO13   109-451   115-514
TPO14   108-432   118-535
TPO15   118-535   118-535   119-536
TPO16   108-432   118-535
TPO17   113-503   114-514
TSBRPE   92-264    93-266
TSCRPE   77-13     92-264
TSTCNT   17-0      76-7      79-31     80-43     81-51     81-52     83-105    84-133    90-250
TSTHNG   77-14     77-14     77-16     77-17     78-22
TSTRTN   84-116    84-117    84-120    84-121    84-124    84-125    85-136    87-164    90-250    91-257   116-524
TSTXL    84-127    85-142
U8K      21-0
UBAR     17-0      95-304    95-311    95-312    98-323    98-327    98-332    98-334    98-335    98-344
UBURST   49-0      49-0
UCRD     18-0      18-0      18-0
UCRS     18-0      18-0      18-0
UCSREN   43-0
UDA1     73-0
UDD      18-0      18-0      18-0
UDDI     17-0      76-3      76-10     77-12
UDS      18-0      18-0      18-0
UER      17-0      80-44     83-74     84-122    84-126    84-131    85-140
UF.576   63-0
UF.CMR   63-0      63-0
UF.CMW   63-0      63-0
UF.MSK   63-0
UF.RMV   63-0
UF.RPL   63-0
UF.WPH   63-0     147-1361  159-2026  159-2030
UF.WPS   63-0     147-1361
UMP      13-0
UMPN     12-0      76-5      76-6
UN.BUF   65-0      65-0     228-7737
UNC.RM   63-0     159-2024
UNC.SS   61-0
UNERR    70-0     156-1712  157-1737
UNITF    46-0
UNLOCK  121-726   153-1573  154-1643  154-1650  159-2012  159-2014  160-2070  162-2111  179-2631  208-3570
UNSUCC   69-0
UP.XFC  228-7760  228-7760  228-7760
UPF      13-0
UPROC    13-0
UQ.CNT   53-0
```



```
URETRY   60-0     123-1001  166-2307  171-2434
UTEMP1   50-0      50-0
UTMP     17-0
UWMCI    40-0
V.ATT2  226-7661  227-7712
V.ATTN  127-1053  226-7655
V.AVAL  128-1054  227-7721
V.CMD1  227-7675
V.DCLR  226-7653  226-7657
V.DCON  153-1607  154-1632  227-7732
V.DRVC  132-1144  226-7643
V.ERRO  171-2453  226-7655
V.ERR1  123-775   227-7705
V.ERRC  158-2003  158-2010  226-7661  226-7661
V.ERSK  158-2003  158-2010  226-7664
V.GCHR  153-1607  227-7735
V.GCR1  155-1657  228-7737
V.GCR2  155-1657  228-7742
V.GSTA  126-1052  227-7727
V.INIT  156-1731  171-2454  226-7654
V.NULL  227-7710  227-7710
V.OLRD  227-7711
V.ONAV  129-1060  227-7724
V.ONLN  129-1057  227-7711
V.RVC3  226-7644
V.RVC5  226-7645
V.RVC8  132-1160  226-7646
V.SEEK  133-1163  150-1513  226-7664  227-7702  227-7704
V.SKON  150-1513  226-7665
V.STAT  153-1607  153-1607  154-1632  227-7731
V.SUCH  134-1164  227-7714
V.TOPO  135-1165  226-7651
V.ZATT  227-7734
VAX.PG   50-0
VAXINT   21-0      21-0
VAXO     48-0
VC.CMD   52-0
VC.DMM   52-0
VC.LOG   52-0
VC.RSP   52-0
VECT     59-0     122-761   122-762   123-776   136-1206  160-2071  162-2111
VECTAB  226-7643
WAIT     80-44     86-153    87-156
WCI      40-0
WD       44-0
WMCI     40-0
WRAP      1-0
WRC      14-0     104-400   184-3022
WRFST    15-0
WRITCD   69-0      69-0      69-0     193-3333
WRITE    40-0
WRNXT    15-0
WRRDX1   69-0      69-0
WRRDXR   69-0
```

KDB50-2.MCR

KDB50-2.MCR

LSCS FORM=FLAG

TIME QUEUED			TIME PRINTED	
			7-SEP-88 08:02	
LAYOUT	**OVERLAY**	**PAPER TYPE**	**FILE TYPE**	**COPIES**
P132D	CONDENSED	3_HOLE	DEFAULT	0001

From: SSDEVO::TEDONE "Matt DTN:522-2254 CX01-2/N26 01-Sep-1988 1533" 1-SEP-1988 17:36
To: GWYNED::GANESAN
Subj: KDB U proc listing (BI stuff)


```
                        ;.SBTTL CONDITIONAL ASSEMBLY EQUATES

                        ;;******************************************************************************
                        ;
                        ;       EQUATES FOR CONDITIONAL ASSEMBLY CODE
                        ;
                        ;;******************************************************************************

000002                  C$TYPE  = 2             ;CONTROLLER TYPE; 0 = UDA, 1 = KDA, 2 = KDB
000000                      $$UDA       = 0
000001                      $$KDA       = 1
000002                      $$KDB       = 2
000000                  LAB     = 0             ;LAB LOOP ON TEST CONTINOUSLY = 1, NOT FOR LAB USE = 0
000000                  WRAP    = 0             ;WRAP TEST = 1 (LAB MUST = 1), NO WRAP TEST = 0
000000                  COPY4   = 0             ;0 = NORMAL 2 COPY HEADER COMPARE, 1 = ALL 4 MUST MATCH
000003                  D.E.C.  = 3
000001                  QDADM   = 1             ;0 = NO DM IN KDA, 1 = DM IN KDA
000001                  QREVB   = 1             ;0 = KDA PRIOR TO REV B, 1 = KDA REV B OR AFTER
000000                  SIMTST  = 0             ;0 = NORMAL CODE, 1 = ASSEMBLE SIMULATE TEST (KDB)
```

```
.PAGE
; UDA CONTROL MEMORY
;       47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32
;     H---+--------H--------+--H-----+--+--H-----------+
;     !CRY!   AD   !   AF   !   AS   !AB!CE!     RB    !
;     +---O--------O--------O--------O--+--+--O--------+
;
;       31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
;     +------------H--------+--H-----------H-----------+
;     !            !        !  !    IOC     !  RA/[BS]  !
;     !     TEST   !   BD   ! @!------------+-----------!
;     !            !        !  !   LITERAL IF AB= 1     !
;     +---O--------O--------O--+-----O--------O--------+
;
;       15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
;     H---+--+-----H----------H----------H-----------+
;     ! S !EN!K !P !         BRANCH ADDRESS (BA)       !
;     +---O--+-----O--------O--------O--------O--------+
; ALU   CRY - AF= 0-2 CARRY INTO ALU
;           - AF= 3-7 HIGH BYTE LITERAL SELECT
;       AD  - ALU DESTINATION    *ALU -> DB IF AB= 1*     *DB= DATA BUS*
;             0= ALU -> Q,DB              4= ALU/2 -> RB, Q/2 -> Q, ALU -> DB
;             1= ALU -> DB                5= ALU/2 -> RB, ALU -> DB
;             2= ALU -> RB, RA -> DB      6= ALUx2 -> RB, Qx2 -> Q, ALU -> DB
;             3= ALU -> RB,DB             7= ALUx2 -> RB, ALU -> DB
;       AF  - ALU FUNCTION    *SELECT HIGH BYTE LITERAL WITH AF= 3-7 AND CRY= 1*
;             0= ADD  R+S       2= REVSUB  R-S  4=  R AND S      6= R XOR  S
;             1= SUB  S-R       3= R OR S       5=  NOT R AND S  7= R XNOR S
;       AS  - ALU SOURCE= R,S *CHOOSE RA OR BS OR LITERAL AS SOURCE*
;             0= RA,Q           2= 0,Q          4= 0,RA          6= D,Q
;             1= RA,RB          3= 0,RB         5= D,RA (RA=RB) 7= D,0 (RA=RB)
; BUS   AB  - ALU TO DATA BUS; 0= BS SRC ON BUS, 1= ALU AS BUS SRC,ENB LITERAL
;       RB  - B PORT REGISTER, ALSO A PORT IF AS= 5 OR 7
;       BD  - BUS DESTINATION
;       RA  - A PORT REGISTER    *[AB= 0/1]*
;     [BS]  - BUS SOURCE         * IF AB= 0*
; IOC   @   - IOC ENABLE         0= DISABLE       1= ENABLE
;       IOC - INPUT OUTPUT CONTROL SELECT (SPECIAL OPERATION)
;       LIT - 8 BIT LITERAL TO LOW  BYTE IF CRY= 0 AND AF= 0-2
;             8 BIT LITERAL TO HIGH BYTE IF CRY= 1 AND AF= 3-7
; BRANCH TEST
;       S   - TEST SENSE:  0= JUMP IF TEST LOW (-),  1= JUMP IF TEST HIGH (+)
;       TEST- TEST CONDITION SELECT                  ,  ELSE  CONTINUE  TO  PC+1
;       EN  - ENABLE CONDITIONAL EXECUTE PER CE BIT 36  0= DISABLE, 1= ENABLE
;       CE  - CONDITIONAL ALU EXECUTE ON TEST RESULT
;             0= EXECUTE IF TEST FALSE (FAILS), 1= EXECUTE IF TEST TRUE(PASS)
;       K   - JUMP OPERATION
;             0= JUMP           1 & BA# OFFFH = JUMP   TO   SUBROUTINE
;                               1 & BA= OFFFH = RETURN FROM SUBROUTINE
;       P   - PARITY BIT FOR ODD PARITY
;       BA  - BRANCH ADDRESS
.PAGE
```

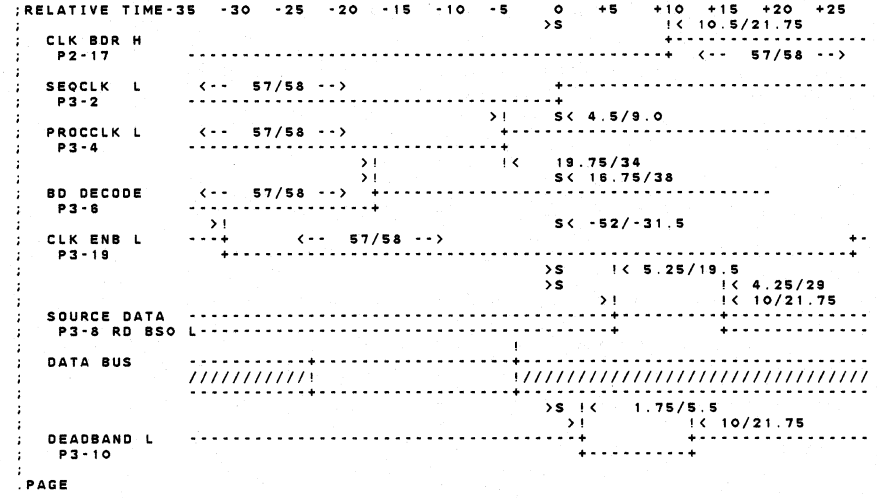


```
; INSTRUCTION FORMAT;
;       LABEL:  ALUOP\mod  intsrc\mod,dst\mod,bd  @ioc  %JUMPOP test,address
;       LABEL:  ALUOP\mod  (extsrc)  ,dst\mod,bd  @ioc  %JUMPTEST address
;               intsrc= 2901 internal source   RA
;               extsrc=      external source  (BS)
;               dst= 2901 internal destination RB
;               bd = 2901 external destination BD
;               mod: \F= execute false     \T= execute on true test condition
;                    \O= output RA         \N= no write to ALU
;                    \L= rotate ALU left   \LQ= shift Q left
;                    \R= rotate ALU right  \RQ shift Q right
;               #LIT= literal source. Can be like ##LIT    ex. <bit00+bit04>
;               #LIT=  lower 8 bit literal with upper 8 bits = zero.
;               ##LIT= upper 8 bit literal with lower 8 bits = zero
; TYPES;        ALU INSTRUCTION FORMAT    SOURCE MODE    OPERATION
;       LABEL:  ALUOP RA,    RB,BD @IOC  ; register      RA OP RB-> RB
;       LABEL:  ALUOP (BS), RB,BD @IOC   ; data bus     (BS)OP RB-> RB, BS->BD
;       LABEL:  ALUOP # LIT,RB,BD @IOC   ; lo/hi literal # LIT OP RB->RB,BD
;       LABEL:  ALUOP ##LIT,RB,BD @IOC   ; upper literal ##LIT OP RB->RB,BD
; MNEMONICS;
;ALUOP= ADD     ADDC    AND     BIC     BIS     BIT     CLR     CMP
;       COM     COMB    DEC     DECB    DECC    INC     INCB    MOV
;       MOVB    NEG     NEGB    NOP     OR      ROT     ROTB    RSUB
;       SUB     SUBC    SHF     SHFB    TST     TSTB    XOR     XNOR
;JUMPOP= JUMP, ADDRESS;          JMP     J       J-
;        CALL, ADDRESS;          JSR     CALL    C       C-
;        RETURN;                 RET     RTN     R       R-
;        TEST (.+1 ALWAYS);      T       T-
;       AZTEC ONLY=     RTNI    JMPI    INT
; INNER INSTRUCTION TIMING RELATIONSHIP:
;       The signal SEQ CLK L determines the start of one instruction from
;       another.  All timing is referanced to this signal.  Branch conditions
;       are tested at the begining of the instruction.  The results of ALU and
;       bus operations are loaded into internal or external registers near the
;       end of the instruction.  The INTERFACE OUTPUT CONTROL (IOC) signals
;       are changed just a little prior to the the register loads.  The
;       signal UPROC H defines which processor is active during the present
;       instruction.  UPROC H = 1 (UPROC L = 0) means the UNIBUS processor is
;       active.  UPROC H = 0 (UPROC L = 1) means the DRIVE processor is
;       active.  A simplified timing relationship is shown below.
;
;       SEQ CLK L       ---+     +----------+     +----------+     +--
;                          +-----+          +-----+          +-----+
; UPROC H        -----------------+                +-----------------+
;                                 +----------------+                 +-----
;  INSTRUCTION   -----------------+----------------+-----------------+-----
;  REGISTER         UNIBUS PROC   !   DRIVE PROC   !   UNIBUS PROC   !
;  INSTRUCTION   -----------------+----------------+-----------------+-----
;                                 ^            ^  ^
;                                 !            !  +-- LOAD REGISTERS
;                                 !            +----- CHANGE IOC'S
;                                 +----------------- TEST BRANCH CONDITION
.PAGE
```

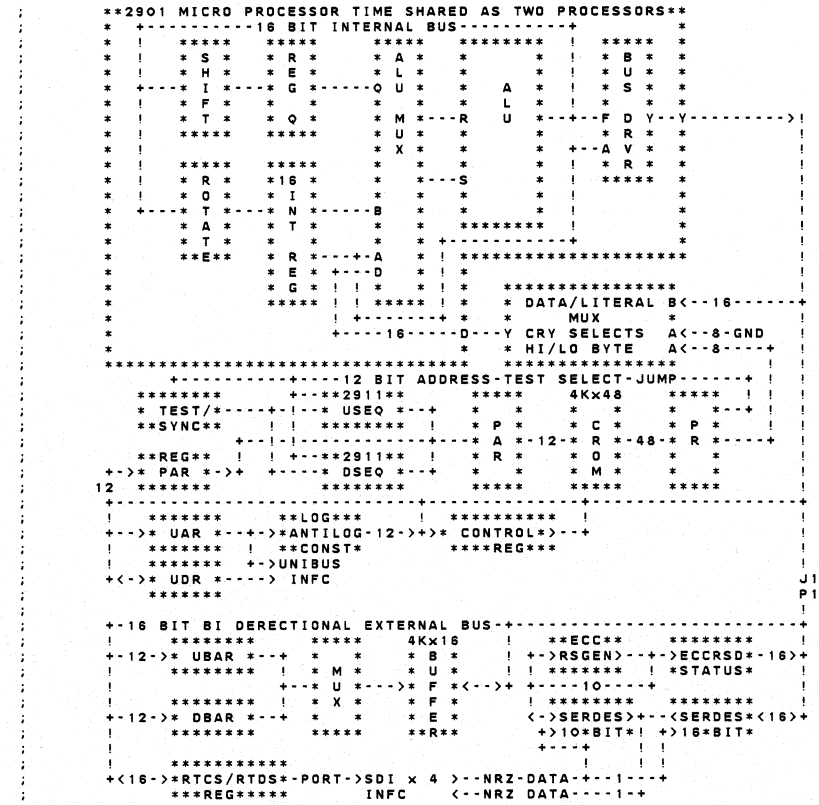
```
; UNIBUS DISK ADAPTER (UDA) TIMING
;
;                 ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++ ++
;                 !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !! !!
;       CLOCK H  -""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-""-
;
;       F/F 1 H  --+58+-116-+  +------+  +------+  +------+  +------+  +------+  +-
;                  +--+     +--+      +--+      +--+      +--+      +--+      +--+
;
;       F/F 2 H  --+      +--+      +--+      +--+      +--+      +--+      +--+
;       CLK ENB H  +-----+58+-116-+  +------+  +------+  +------+  +------+  +----
;
;                --+      +--+      +--+      +--+      +--+      +--+      +--+
;(K.SDI CLOCK IN H)+-----+58+-116-+  +------+  +------+  +------+  +------+  +----
;
;                  +-----+58+-116-+  +------+  +------+  +------+  +------+  +----
;       SEQ CLK L -+      +--+      +--+      +--+      +--+      +--+      +--+
;
;                  +--174---+        +---------+        +---------+        +----
;       UPROC  H --+        +--174---+         +---------+         +---------+
;
;                --+--------+--------+--------+--------+--------+--------+----
;       SEQ ADR H  !/////U+1!/////D+1!/////U+2!/////D+2!/////U+3!/////D+3!
;                --+--------+--------+--------+--------+--------+--------+----
;
;       PIPELINE --+--------+--------+--------+--------+--------+--------+----
;       ADDRESS REG!   D    !  U+1   !  D+1   !  U+2   !  D+2   !  U+3   !
;       PADR    H --+--------+--------+--------+--------+--------+--------+----
;
;       PIPELINE --+--------+--------+--------+--------+--------+--------+----
;       REG=CURRENT!   U    !   D    !  U+1   !  D+1   !  U+2   !  D+2   !
;       INSTRUCTION+--------+--------+--------+--------+--------+--------+----
;
.PAGE
```



```
; RELATIVE CLOCK TIMING - GENERAL
;
; Common K.SDI, K.DISK, K.???, UDA Terminology
;       1. Prefix all clocks with CLK, all bus sources with RD, all bus
;          destinations with LD.
;       2. Have common page order ie. ALU, CLOCKING, BRANCH, SEQ, PAR, CROM etc.
;
;K.DISK= CLOCK IN H-+                   +---------+                   +-------
;UDA=    CLK ENB H  +-------------------+         +-------------------+
;                       !                             !
;                   -+                   +---------+                   +------
;       CLK ENB1 H   +-------------------+         +-------------------+
;                       !                             !
;                   --+                   +---------+                   +-----
;       CLK ENB2 H    +-------------------+         +-------------------+
;                       !                             !
;                   ---+                   +---------+                   +----
;       CLK ENB3 H     +-------------------+         +-------------------+
;                       !                             !
;                   ----+                   +---------+                   +---
;       CLK ENB4 H      +-------------------+         +-------------------+
;                       !                             !
;                   -----+                   +---------+                   +--
;       CLK ENB5 H       +-------------------+         +-------------------+
;                       !                             !
;                       +-------------------+         +-------------------+
;       SEQ CLK L ------+                   +---------+                   +---
;                       !                             !
;                        +-----------------------------+
;       UPROC H   -------+                             +----------------------
;                       !                             !
;                                           +-----------------------------+
;       DLY UPROC H ------------------------+                             +---
;                       !                             !
;                      +-------------------+         +-------------------+
;       PROCCLK L -----+                   +---------+                   +----
;                       !                             !
;                     +-------------------+         +-------------------+
;       LD CLK L  ----+                   +---------+                   +-----
;                       !                             !
;                     +-------------------+         +-------------------+
;       BUFF WE   ----+                   +---------+                   +-----
;                       !                             !
;                        +-------------------+         +-------------------+
;       CLK BDR H -------+                   +---------+                   +--
;                       !                             !
;                 ------+-+---------------------------+-+---------------------
;                       !/!                           !/!
;       RD CLK L  ------+-+---------------------------+-+---------------------
;
.PAGE
```



.PAGE



.PAGE

```
                        .SBTTL  KDB MICROCODE
                        .SBTTL  HARDWARE/SOFTWARE DEFINITIONS AND EQUATES

                        ; *-----------------------------------------------------------------------*
                        ; *     HARDWARE/SOFTWARE DEFINITIONS AND EQUATES                         *
                        ; *     14-JAN-1985      UDA00.UQA                                        *
                        ; *-----------------------------------------------------------------------*
                        ;
                        ;REVISION HISTORY:
                        ;
```



```
                        ;       PAGE ADDRESS   ADR*6=    WAS     SHOULD BE  TYPE          DESCRIPTION
                        ;                      PATCH                        CHANGE
                        ;             (OCT)    (OCT)    (OCT)    (OCT)      PERM TEMP
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
                        ;      !    !        !        !        !          !    !    !                          !
                        ;      +----+--------+--------+--------+----------+----+----+--------------------------+
```

```
;*-----------------------------------------------------------------------------*
;*      USING THE 2901 ALU AS A COMPARITOR                                     *
;*-----------------------------------------------------------------------------*
;
;                          LOGICAL OPERATIONS
;   FUNCTION  OPERATION          TEST CONDITION              RESULT
;    RA = 0      MOV      RA,RB   ZRO =    TRUE            RB= RA
;    RA # 0      MOV      RA,RB   ZRO =    FALSE             '
;    RA = RB     XOR      RA,RB   ZRO =    TRUE            RB= RA XOR  RB
;    RA # RB     XNOR     RA,RB   ZRO =    FALSE           RB= RA XNOR RB
;                         ARITHMETIC OPERATIONS - CARRY IN OFF
;    SAME SIGN   XOR      RA,RB\N NEG =  TRUE       -    RA XOR RB -> BD  P-P,N-N
;    DIFF SIGN   XOR      RA,RB\N NEG =  FALSE      -    RA XOR RB -> BD  P-N,P-N
;                                      P-P,N-N   P-N,N-P
;    RA < RB     SUBC     RA,RB   CRY =  TRUE     TRUE   RB-RA-1-> RB,BD  (BORROW)
;    RA>= RB     SUBC     RA,RB   CRY =  FALSE (TRUE)      '
;    RA > RB    (RSUBC)   RA,RB   CRY =  TRUE     FALSE  RA-RB-1-> RB,BD  (BORROW)
;    RA<= RB    (RSUBC)   RA,RB   CRY =  FALSE    TRUE             '
;                         ARITHMETIC OPERATIONS - CARRY IN ON
;                                      P-P,N-N   P-N,N-P
;    RA<= RB     SUB      RA,RB   CRY =  TRUE     FALSE  RB-RA ->  RB,BD  (BORROW)
;    RA > RB     SUB      RA,RB   CRY =  FALSE    TRUE             '
;    RA = RB     SUB      RA,RB   ZRO =  TRUE     TRUE   RB-RA ->  RB,BD   (BORROW)
;    RA # RB     SUB      RA,RB   ZRO =  FALSE    FALSE            '
;                                      P-P,N-N   P-N,N-P
;    RA>= RB     RSUB     RA,RB   CRY =  TRUE     FALSE  RA-RB ->  RB,BD   (BORROW)
;    RA < RB     RSUB     RA,RB   CRY =  FALSE    TRUE             '
;    RA>= RB     CMP      RA,RB   CRY =  TRUE     FALSE  RA-RB ->      BD   (BORROW)
;    RA < RB     CMP      RA,RB   CRY =  FALSE    TRUE             '
;    RA = RB     RSUB     RA,RB   ZRO =  TRUE     TRUE   RA-RB ->  RB,BD   (BORROW)
;    RA # RB     RSUB     RA,RB   ZRO =  FALSE    FALSE            '
;
;   NOTE:     ADDITION  CRY = TRUE   MEANS ADD    ONE TO   NEXT HIGHER 16 BITS
;          SUBTRACTION CRY = FALSE  MEANS BORROW ONE FROM NEXT HIGHER 16 BITS
;              RA= REGISTER A SOURCE   RB= REGISTER B SOURCE,DESTINATION
;              RB\N=   REGISTER B SOURCE, NO LOAD
;              P-P= POSITIVE MINUS POSITIVE     P-N= POSITIVE MINUS NEGATIVE
;              N-N= NEGATIVE MINUS NEGATIVE     N-P= NEGATIVE MINUS POSITIVE
;          RA>= RB <==> RB<= RA          RA < RB <=> RB > RA
;
;   COMPARITOR EXCEPTIONS;
;       1. MAXIMUM POSITIVE NUMBER IS 32767 (HIGHER CHANGES THE SIGN BIT)
;       2. DIFFERENT SIGNS IMPLIES AN EQUATE IS NOT POSSIBLE EXCEPT
;          FOR ZERO SINCE IT HAS THE SAME VALUE POSITIVE OR NEGATIVE.
```



```
;*-----------------------------------------------------------------------------*
;*      PROCESSOR RESTRICTIONS - SUMMARY                                       *
;*-----------------------------------------------------------------------------*
;       PAR REGISTER - When loading the PAR register, neither source may be
;       changed to allow the address to be stable to the very end of the cycle.
;       2. Loading the PAR register must be unconditional to allow the pull-up
;       resistors enough time to get above the threshold voltage.
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *      TEST CONDITIONS FOR "U" PROCESSOR                                  *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ;       THE TEST CONDITION EQUATES ARE OF THE FOLLOWING FORM:
                        ;               BIT4    POSITIVE TRUE CONDITION 0= FALSE, 1= TRUE (CROM15)
                        ;               BIT4    NEGATIVE TRUE CONDITION 1= FALSE, 0= TRUE (CROM15)
                        ;               BIT0-3  TEST CONDITION SELECT (CROM28-31)
                        ;       DEFINITIONS ARE IN OCTAL UNLESS OTHERWISE SPECIFIED
                        ; *-------------------------------------------------------------------------*
                        ; *              BUS PROCESSOR NEGATIVE TRUE CONDITION DEFINITION           *
                        ; *-------------------------------------------------------------------------*

000000                  UMPN    := 0            ; xDA - JUMP NOT = CONTINUE
000001                  NZRO    := 1            ; ALU - NOT = ZERO (NOT EQUALS)
000001                  NEQ     := 1            ; ALU - SRC NOT = DST
000002                  NCRY    := 2            ; ALU - CARRY OUT FALSE
000002                  LESS    := 2            ; ALU - SRC IS LESS THAN DST
000003                  NMSB    := 3            ; ALU - MSB =0
000003                  NNEG    := 3            ; ALU - COMPARITOR MSB = 0,NOT NEGATIVE (POS OR ZERO)
000004                  LSB     := 4            ; ALU - LSB =1
000005                  TEST    := 5            ; xDA - EXT.TEST L (P3-3) ACTIVE, DO EXTRA TEST CODE
000006                  NUPF    := 6            ; xDA - FLAG RESET BY IOC OF U PROCESSOR @RUPF
000006                  DPROC   := 6            ; xDA - SPLIT ; @RUPF -> DPROC JUMP/UPROC CONTINUE (PC+1)
000007                  ACLO    := 7            ; xDA - AC POWER FAILING
000007                  nrpok   := 7    ;temp   ; xDA - not power ok [qda compat]
000010                  FTEST   := 10           ; xDA - FAST SELF TEST
000011                  STOP    := 11           ; xDA - STOP COMMAND ISSUED - NO MORE BI COMMANDS
000012                  CPE     := 12           ; xDA - CONTROL MEMORY PARITY ERROR
000013                  ETST1   := 13           ; xDA - EXTERNAL TEST 1
000014                  MERR    := 14           ; xDA - BI COMMAND ERROR ON MASTER PORT (DURING XFER)
000015                  SCAN    := 15           ; xDA - IP ACCESSED
000015                  POLL    := SCAN
000016                  CDONE   := 16           ; xDA - COMMAND DONE
000017                  NBIBAD  := 17           ; xDA - BI BAD L SIGNAL NOT ASSERTED LOW (TBAD H IS LOW)
```



```
                        ; *-------------------------------------------------------------------------*
                        ; *              BUS PROCESSOR POSITIVE TRUE CONDITION DEFINITION           *
                        ; *-------------------------------------------------------------------------*

000020                  UMP     := 20           ; xDA - JUMP UNCONDITIONAL
                        ;JMP    := 20           ; ASSEMBLER DEFINED - JMP IS UNCONDITIONAL JUMP
000021                  ZRO     := 21           ; ALU - EQUALS ZERO
000021                  EQ      := 21           ; ALU - SRC = DST
000022                  CRY     := 22           ; ALU - CARRY OUT TRUE
000022                  GTE     := 22           ; ALU - SRC IS GREATER THAN OR SAME AS DST
000023                  MSB     := 23           ; ALU - MSB = 1
000023                  NEG     := 23           ; ALU - RESULT NEGATIVE          MSB= 1
000024                  NLSB    := 24           ; ALU - LSB = 0
000025                  NTEST   := 25           ; xDA - EXT.TEST L (P3-3) INACTIVE
000026                  UPF     := 26           ; xDA - FLAG SET BY IOC OF BUS PROCESSOR @SUPF
000026                  UPROC   := 26           ;       @SUPF -> UPROC JUMP / DPROC CONTINUE (PC+1)
000027                  NACLO   := 27           ; xDA - AC POWER NOT FAILING
000027                  rpok    := 27   ;temp   ; xDA - power ok [qda compat]
000030                  NFTEST  := 30           ; xDA - SLOW SELF TEST
000030                  STEST   := NFTEST
000031                  NSTOP   := 31           ; xDA - STOP COMMAND NO ISSUED - MORE BI COMMANDS
000032                  NCPE    := 32           ; xDA - NO CONTROL MEMORY PARITY ERROR
000033                  NETST1  := 33           ; xDA - NOT EXTERNAL TEST 1
000034                  NMERR   := 34           ; xDA - NO BI COMMAND ERROR ON MASTER PORT (DURING XFER)
000035                  NSCAN   := 35           ; xDA - NO IP ACCESS
000035                  NPOLL   := NSCAN
000036                  NCDONE  := 36           ; xDA - NOT COMMAND DONE
000037                  BIBAD   := 37           ; xDA - BI BAD L IS ASSERTED LOW (TBAD H IS HIGH)
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *     TEST CONDITIONS FOR "D" PROCESSOR (WHERE DIFFERENT ONLY)             *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ; *-------------------------------------------------------------------------*
                        ; *              DRIVE PROCESSOR NEGATIVE TRUE CONDITION DEFINITION         *
                        ; *-------------------------------------------------------------------------*

000006                  NDPF    := 6            ; xDA - D PROCESSOR FLAG NOT SET (@RDPF)
000010                  ndclk   := 10   ;temp   ; sdi - serdes parallel data not ready
000010                  NPRDY   := 10           ; SDI - SERDES PARALLEL DATA NOT READY
000011                  NCSR    := 11           ; SDI - CONTROL STATE NOT READY - RTCS NOT READY TO UPDATE (RELOAD)
000012                  SECTR   := 12           ; SDI - SECTOR PULSE
000012                  nrrdy   := 12   ;temp   ; serdes - ecc residue parallel data not ready [qda compat]
000013                  RPE     := 13           ; xDA - RAM PARITY ERROR
000014                  NDSER   := 14           ; NOT REAL TIME DRIVE STATE ERROR
000015                  NDSR    := 15           ; DRIVE STATE NOT READY - NEW RTDS NOT RECEIVED SINCE LAST WRITTEN TO RTCS
000016                  nlate   := 16   ;temp   ; sdi - no serdes over/under runs
000016                  NOVER   := 16           ; SDI - NO SERDES OVER/UNDER RUNS
000017                  NWRC    := 17           ; xDA - WORD RATE CLOCK LOW

                        ; *-------------------------------------------------------------------------*
                        ; *              DRIVE PROCESSOR POSITIVE TRUE CONDITION DEFINITION         *
                        ; *-------------------------------------------------------------------------*

000026                  DPF     := 26           ; xDA - D PROCESSOR FLAG SET (@SUPF)
000030                  dclk    := 30   ;temp   ; sdi - serdes parallel data ready
000030                  PRDY    := 30           ; SDI - SERDES PARALLEL DATA READY
000031                  CSR     := 31           ; SDI - CONTROL STATE READY - RTCS READY TO UPDATE (RELOAD)
000032                  NSECTR  := 32           ; SDI - NO SECTOR PULSE
000032                  rrdy    := 32   ;temp   ; serdes - ecc residue parallel data ready [qda compat]
000033                  NRPE    := 33           ; xDA - RAM PARITY ERROR
000034                  DSER    := 34           ; SDI - REAL TIME DRIVE STATE ERROR (PARITY,PULSE, OR DATA)
000035                  DSR     := 35           ; DRIVE STATE READY - NEW RTDS RECEIVED SINCE LAST WRITTEN TO RTCS
000036                  late    := 36   ;temp   ; sdi - serdes over/under run has occurred
000036                  OVER    := 36           ; SDI - SERDES OVER/UNDER RUN HAS OCCURRED
000037                  WRC     := 37           ; xDA - WORD RATE CLOCK HIGH
```



```
                        ;       *---------------------------------------------------------*
                        ;       *                INPUT/OUTPUT CONTROL REGISTER            *
                        ;       *---------------------------------------------------------*
                        ;
                        ;       EACH PROCESSOR (BUS OR DRIVE) HAS AN ADDRESSABLE LATCH
                        ;       WHICH IS USED FOR INPUT/OUTPUT CONTROLS.  SOME BITS HAVE MULITPLE
                        ;       USES.  EACH SIGNAL IS INDIVIDUALLY SET OR RESET IF ENABLED BY THE
                        ;       CURRENT MICROINSTRUCTION.
                        ;               BIT3    0= SIGNAL TO LOW LEVEL, 1= SIGNAL TO HIGH (CROM23)
                        ;               BIT0-2  IOC SELECT (CROM20-22)
                        ;
                        ; *-------------------------------------------------------------------------*
                        ; *              BUS PROCESSOR IOC SIGNAL DEFINITIONS                       *
                        ; *-------------------------------------------------------------------------*

000000                  SUPF    := 0            ; xDA - UTEST H SET
000001                  GORD    := 1            ; xDA - GO_READ
000002                  LNPAG   := 2            ; xDA - LOAD NEXT PAGE ADDRESS LOWER
000003                  WRSND   := 3            ; xDA - WRITE SECOND BUFFER
                        ;LOFF   := 3            ; xDA - LOAD ADDRESS BUFFER OFFSET
000004                  RDFST   := 4            ; xDA - READ FIRST BUFFER
000005                  WRFST   := 5            ; xDA - WRITE FIRST BUFFER
000006                  LCOM    := 6            ; xDA - LOAD COMMAND
000007                  LADD    := 7            ; xDA - LOAD COMAND ADDRESS LOWER

000010                  RUPF    := 10           ; xDA - UTEST H CLR
                        ;SCLR   := 11           ; xDA - CLEAR CROM PE (defined in UDIAG.UQA)
                        ;RCLR   := 11   ;temp   ; xDA - CLEAR CROM PE (defined in UDIAG.UQA) [qda compat]
                        ;WRBYT  := 12           ; xDA - WRITE BYTE ON NEXT OPERATION
000012                  GOWR    := 12           ; xDA - GO WRITE
000013                  RBCAI   := 13           ; xDA - RESET BCAI
000014                  RDNXT   := 14           ; xDA - READ NEXT BUFFER
000015                  WRNXT   := 15           ; xDA - WRITE NEXT BUFFER
000016                  LBWR    := 16           ; xDA - WRITE LAST BUFFER AND GO_WRITE
000017                  HADD    := 17           ; xDA - ADDRESS
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *              DRIVE PROCESSOR IOC SIGNAL DEFINITIONS                     *
                        ; *-------------------------------------------------------------------------*
                        ;       CERTAIN SIGNALS MUST BE SEQUENCED IN ORDER FOR THE HARDWARE TO WORK:
                        ;       ?       WRITING DATA            FIRST @RCMD     THEN @SSE
                        ;       ?       WRITING COMMANDS        FIRST @SCMD     THEN @SSE
                        ;       ?       READING DATA            FIRST @RCMD     THEN @SSE
                        ;       ?       READING COMMANDS        FIRST @RCMD     THEN @SSE
                        ;       ?       DEFAULT                 FIRST @SCMD     THEN @SSE
                        ;
000000                  SDPF    := 0            ; xDA - SET DRIVE PROCESSOR FLAG (TEST AS COMPLIMENT)
000001                  RRSGEN  := 1            ; SDI - RESET RSGEN CLOCK. THIS WILL ALLOW RSGEN CLOCKS TO OCCUR
                                                ;       FOLLOWING A @SRSGEN A @RRSGEN WILL UPDATE RSGEN AND ECC REGISTERS
000002                  SCMD    := 2            ; SDI - SET SDI LEVEL 2 COMMAND
                        ;SCLR   := 3            ; SDI - SET SDI INFC I/O CLEAR (defined in DDIAG.UQA)
000004                  RSE     := 4            ; SERDES - RESET SERDES ENABLE/-RESET REG,CNT    MR0,CE
000005                  RRM     := 5            ; SERDES - RESET READ MODE                 SDIC,POM
000005                  SWM     := 5            ; SERDES - SET WRITE MODE
000006                  RECC    := 6            ; SERDES - RESET ECC ENABLE
000007                  RECCT   := 7            ; SERDES - RESET ECC TIME

000010                  RDPF    := 10           ; xDA - RESET D PROCESSOR FLAG (TEST AS COMPLIMENT)
000011                  SRSGEN  := 11           ; SDI - SET RSGEN CLOCK. THIS WILL FORCE RSGEN CLOCK TO BE LOW
000012                  RCMD    := 12           ; SDI - FOR ALL OTHER FUNCTIONS
                        ;RCLR   := 13           ; SDI - RESET SDI INFC I/O CLEAR (defined in DDIAG.UQA)
000014                  SSE     := 14           ; SERDES - SET SERDES ENABLE/-RESET REG,CNT      MR0,CE
000015                  SRM     := 15           ; SERDES - SET READ MODE
000015                  RWM     := 15           ; SERDES - RESET WRITE MODE
000016                  SECC    := 16           ; SERDES - SET ECC ENABLE
000017                  SECCT   := 17           ; SERDES - SET ECC TIME
```



```
                        ; *-------------------------------------------------------------------------*
                        ; *                    2901 INTERNAL REGISTER SUMMARY                       *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ; These are registers which are internal to the 2901 bit slice chip. They
                        ; consist of 16 Dual Port RAM registers and 1 accumulator (Q) register. The 16
                        ; RAM registers are configured for a 16 bit rotate while the Q register is a 16
                        ; bit shift.
                        ;
                        ;MNEMONIC        DESCRIPTION                                          UPROC   DPROC
                        ;       OCTAL ADDRESS
                        ;
                        ;       := Q     ; BUS PROCESSOR TIMER/COUNTER REGISTER  R/W          -
                        ;       := R0    ; BUS PROCESSOR TEMPORARY REGISTER                   R/W     -
                        ;       := R1    ; BUS PROCESSOR TEMPORARY REGISTER                   R/W     -
                        ;       := R2    ; BUS PROCESSOR TEMPORARY REGISTER                   R/W     -
                        ;       := R3    ; BUS PROCESSOR TEMPORARY REGISTER                   R/W     -
000004                  RLL     := R4    ; RICHY LARY'S LOCK AND SYSTEM STATUS                R/W     R/W
000004                  CRI     := R4    ; *** TEMP UNTIL MATT'S CODE CHANGED
000005                  UBAR    := R5    ; BUS PROCESSOR BUFFER ADDRESS REG IMAGE             R/W     -
000006                  IUAR    := R6    ; BUS ADDRESS REGISTER IMAGE                         R/W     -
000007                  UER     := R7    ; UPROC ERROR REGISTER                               R/W     -
000010                  UTMP    := R10   ; UPROC TEMPORARY REGISTER                           R/W     -
000011                  DER     := R11   ; DPROC ERROR REGISTER                               -       R/W
                        ;       := R11   ; DRIVE PROCESSOR TIMER/COUNTER REGISTER             -       R/W
000012                  DTMP    := R12   ; DPROC TEMPORARY REGISTER                           -       R/W
000012                  RPC     := R12   ; DM INTERPRETER PC                                  -       R/W
                        ;       := R12   ; DRIVE PROCESSOR TEMPORARY REGISTER                 -       R/W
000013                  RIB     := R13   ; DM INTERPRETER INSTRUCTION BUFFER                  -       R/W
                        ;       := R13   ; DRIVE PROCESSOR TEMPORARY REGISTER                 -       R/W
000014                  RCC     := R14   ; DM INTERPRETER ARITH/LOGICAL CONDITION CODE        -       R/W
                        ;       := R14   ; DRIVE PROCESSOR TEMPORARY REGISTER                 -       R/W
000015                  DBAR    := R15   ; DRIVE PROCESSOR BUFFER ADDRESS REGISTER IMAGE      -       R/W
000015                  RCY     := R15   ; DM INTERPRETER LAST ARITH OP CARRY                 -       R/W
000016                  RT1     := R16   ; DM INTERPRETER GENERAL PURPOSE TEMP                -       R/W
                        ;       := R16   ; DRIVE PROCESSOR TEMPORARY REGISTER                 -       R/W
                        ;       := R17   ; DRIVE PROCESSOR TEMPORARY REGISTER                 -       R/W

                        ; *-------------------------------------------------------------------------*
                        ; *                2901 INTERNAL REGISTER SUMMARY - DIAGNOSTICS             *
                        ; *-------------------------------------------------------------------------*

                        ;MNEMONIC        DESCRIPTION                                          UPROC   DPROC
                        ;       OCTAL ADDRESS

000000                  TSTCNT  := Q     ; BUS PROCESSOR TIMER/COUNTER REGISTER               R/W     -
000003                  HANG    := R3    ; BUS PROCESSOR TEMPORARY REGISTER                   R/W     -
000010                  UDDI    := R10   ; BUS DATA REGISTER REGISTER IMAGE                   R/W     -
```

```
                        ;*------------------------------------------------------------------------------*
                        ;*                    2901 EXTERNAL HARDWARE REGISTER SUMMARY                   *
                        ;*------------------------------------------------------------------------------*
                        ;
                        ;MNEMONIC       DESCRIPTION
                        ;                                  #1= On xDA#1 PC BOARD.
                        ;     OCTAL ADDRESS                #2= On xDA#2 PC BOARD.                  UPROC   DPROC

000000                  NOPREG  := 0     ; (NON EXISTANT)                                          WRITE   WRITE
000002                  RTDS    := 2     ; NO OPERATION REGISTER (NON EXISTANT)                    (RD)    READ
000002                  PAR     := 2     ; PIPELINE ADDRESS REGISTER                               WRITE#1 WRITE #1
000003                  BUF     := 3     ; RAM DATA BUFFER                                         R/W #2  R/W #2
000004                  CR      := 4     ; DPROC - CONTROL REGISTER SOURCE                          -      R/W #2
                                         ; UPROC - CONTROL REGISTER DESTINATION                    WRITE#1
000014                  UCRS    := 14    ; UPROC - CONTROL REGISTER SOURCE                         READ#1  (RD)
000004                  UCRD    := 4     ; UPROC - CONTROL REGISTER DESTINATION                    WRITE#1
000001                  DCRS    := 1     ; DPROC - CONTROL REGISTER SOURCE                                 READ#2
000004                  DCRD    := 4     ; DPROC - CONTROL REGISTER DESTINATION                            WRITE#2
000005                  RTCS    := 5     ; DPROC - REAL TIME CONTROLLER STATE                       -      WRT#2
000015                  BCAIS   := 15    ; UPROC - BCAI REGISTER SOURCE                            READ #1 (RD)
000005                  BCAID   := 5     ; UPROC - BCAI REGISTER DESTINATION                       WRITE#1 (RD)
000015                  UDS     := 15    ; UPROC - BUS DATA REGISTER SOURCE      [qda compat]      READ #1 (RD)
000005                  UDD     := 5     ; UPROC - BUS DATA REGISTER DESTINATION [qda compat]      WRITE#1 (RD)
000006                  BREG    := 6     ; UPROC - BI REGISTER                                     WRITE #1
000006                  SD      := 6     ; DPROC - SERDES 16 DATA REGISTER                         (RD)    R/W #2
000007                  ECC     := 7     ; DPROC - RS GENERATOR REGISTER (RESIDUE)                 (RD)    READ#2
000007                  BAR     := 7     ; UPROC - BUFFER ADDRESS REGISTER                         WRITE#2  -
                                         ; DPROC - BUFFER ADDRESS REGISTER                          -      WRT #2
000010                  FLOAT   := 10    ; NO BUS SOURCE REGISTER & BUS FLOAT                      READ    READ
                        ;*------------------------------------------------------------------------------*
                        ;*      BIT DEFINITIONS                                                         *
                        ;*------------------------------------------------------------------------------*

100000                  BIT15   := 100000
040000                  BIT14   := 40000
020000                  BIT13   := 20000
010000                  BIT12   := 10000
004000                  BIT11   := 4000
002000                  BIT10   := 2000
001000                  BIT09   := 1000
000400                  BIT08   := 400
000200                  BIT07   := 200
000100                  BIT06   := 100
000040                  BIT05   := 40
000020                  BIT04   := 20
000010                  BIT03   := 10
000004                  BIT02   := 4
000002                  BIT01   := 2
000001                  BIT00   := 1
```



```
                        ;*------------------------------------------------------------------------------*
                        ;*                    2901 INTERNAL REGISTER DEFINITIONS                        *
                        ;*------------------------------------------------------------------------------*
                        ;
                        ;MNEMONIC  ADDRESS       DESCRIPTION                                  UPROC   DPROC
                        ;
                        ;RLL     := R4           ; RICHY LARY LOCK & STATUS REGISTER          R/W     R/W
                        ;
000001                  BLOCK   := BIT00         ; BUFFER LOCK FLAG, 0 = LOCKED, 1 = FREE
                        ;       := BIT01         ; BUFFER STACK POINTER
                        ;       := BIT02         ; BUFFER STACK POINTER
                        ;       := BIT03         ; BUFFER STACK POINTER
                        ;       := BIT04         ; BUFFER STACK POINTER
                        ;       := BIT05         ; BUFFER STACK POINTER
                        ;       := BIT06         ; BUFFER STACK POINTER
                        ;       := BIT07         ; BUFFER STACK POINTER
                        ;
                        ;       := BIT08         ; BUFFER STACK POINTER
001000                  INDIAG  := BIT09         ; 0 = FUNCTIONAL CODE, 1 = xDA DIAGNOSTIC CODE RUNNING
                        ;       := BIT10         ; NOT USED
004000                  CN.ERR  := BIT11         ; 0 = NO ACTION, 1 = D.PROC FATAL CONTROLLER ERROR (CODE IN R11)
010000                  DXFC    := BIT12         ; 0 = NO ACTION, 1 = U.PROC XFC SERVICE REQUIRED
020000                  DMBEG   := BIT13         ; 0 = NO ACTION, 1 = START DM INTERPRETER (START ADDRESS IN RPC)
040000                  DMODE   := BIT14         ; 0 = NORMAL MODE, 1 = DIAGNOSTIC MACHINE MODE
100000                  PLOCK   := BIT15         ; PROCESSOR LOCK FLAG, 0 = LOCKED, 1 = FREE
                        ;
                        ;UER     := R7           ; UPROC ERROR REGISTER                        R/W      -
                        ;DER     := R11          ; DPROC ERROR REGISTER                         -       R/W
                        ;
                        ;       THE ERROR DISPLAY IS BIT ENCODED TO CORRESPOND TO THE ERROR MESSAGE
                        ;
                        ;       *** SEE BUS I/O REGISTER DEFINITIONS    ***
                        ;
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *                    2901 EXTERNAL REGISTER DEFINITIONS                   *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ;MNEMONIC  ADDRESS      DESCRIPTION                                  UPROC   DPROC
                        ;
                        ;NOPREG  := 00          ; NO OPERATION REGISTER (NON EXISTANT)       R/W     R/W
                        ;
                        ;        := BIT00       ; + NON   EXISTANT
                        ;        := BIT01       ; +   '       '
                        ;        := BIT02       ; +   '       '
                        ;        := BIT03       ; +   '       '
                        ;          --------
                        ;        := BIT04       ; +   '       '
                        ;        := BIT05       ; +   '       '
                        ;        := BIT06       ; +   '       '
                        ;        := BIT07       ; +   '       '
                        ;          --------
                        ;        := BIT08       ; +   '       '
                        ;        := BIT09       ; +   '       '
                        ;        := BIT10       ; +   '       '
                        ;        := BIT11       ; +   '       '
                        ;          --------
                        ;        := BIT12       ; +   '       '
                        ;        := BIT13       ; +   '       '
                        ;        := BIT14       ; +   '       '
                        ;        := BIT15       ; +   '       '
```



```
                        ;MNEMONIC  ADDRESS      DESCRIPTION                                  UPROC   DPROC
                        ;
                        ;UCRS    := 14          ; UPROC CONTROL REGISTER SOURCE              READ      -
                        ;UCRD    := 04          ; UPROC CONTROL REGISTER DESTINATION         WRITE     -
                        ;
                        ;       THIS REGISTER IS READ/WRITE.  IT IS DIFFERENT FROM xDA#2 CR REGISTER
                        ;       THAT THE LED'S ARE ALWAYS ON. DIAGNOSTIC MODE HAS NO EFFECT ON ANY
                        ;       xDA#1 HARDWARE.
                        ;
                        ;        := BIT00       ; LSB WORD COUNT BIT
                        ;        := BIT01       ;
                        ;        := BIT02       ;
                        ;        := BIT03       ; MSB WORD COUNT BIT
                        ;          -------
                        ;        := BIT04       ; RESERVED
                        ;        := BIT05       ; RESERVED
000100                  U8K      := BIT06       ; 0 = ENABLE DPROC FAIL TEST, 1 = ENABLE UPROC FAIL TEST
                        ;OPM     := BIT07       ; DIAGNOSTIC MODE L  *THIS BIT DOES NOTHING TO THE HARDWARE xDA#1*
                        ;          -------
000400                  vaxint   := BIT08  ;temp ; 1 = vax kludge interrupt in progress [uda compat]
                        ;        := BIT09       ; NOT USED
002000                  DFAIL    := BIT10       ; 0 = ENABLE DPROC SEQUENCER,1= FORCE DPROC TO FAIL TEST
004000                  DRINIT   := BIT11       ; 1 = DRIVE INIT IN PROGRESS (USED FOR SOFTWARE INIT)
                        ;          -------
                        ;        := BIT12       ; LED 1 L, 0 = TURN LED ON, 1 = TURN LED OFF
                        ;        := BIT13       ; LED 2 L       "               "
                        ;        := BIT14       ; LED 4 L       "               "
                        ;        := BIT15       ; LED 8 L       "               "
                        ;
```

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                          UPROC   DPROC
                                ;
                                ;DCRS    := 01          ; DPROC CONTROL REGISTER SOURCE                       -      READ
                                ;DCRD    := 04          ; DPROC CONTROL REGISTER DESTINATION                  -      WRITE
                                ;
                                ;                 NOTE - 1. IN DIAGNOSTIC MODE THE D PROCESSOR SHOULD MOVE CRI
                                ;                 TO CR AND WAIT FOR THE LEAST SIGNIFICANT(some) BIT TO COME
                                ;                 ON BEFORE PROCEEDING.  THIS WAY THE U PROCESSOR CAN ENABLE
                                ;                 OR DISABLE RAM PARITY ERROR AND FLASH THE LIGHTS FOR
                                ;                 BOTH BOARDS.  LEAST SIGNIFICANT BIT = 0 MEANS WAIT, = 1
                                ;                 MEANS CONTINUE.
                                ;                 2. IN DIAGNOSTIC MODE THERE IS A 1 BIT DELAY FROM DIAG DATA TO
                                ;                 NRZ DATA IN AND A 21 BIT DELAY FROM LOADING RTCS TO VALID RTDS.
                                ;
                                ;        := BIT00         ; 1 = SELECT DRIVE PORT 1
                                ;        := BIT01         ; 1 = SELECT DRIVE PORT 2
                                ;        := BIT02         ; 1 = SELECT DRIVE PORT 3
                                ;        := BIT03         ; 1 = SELECT DRIVE PORT 4
                                ;          -------
000020                          FDCLK    := BIT04         ; USE DIAGNOSTIC CLOCK=0,USE FAST CLOCK=1
000040                          SERIAL   := BIT05         ; SERIAL PORT
000100                          RAMPE    := BIT06         ; 0 = INHIBIT RAM PARITY ERROR, 1 = ENABLE RAM PARITY ERROR
000200                          OPM      := BIT07         ; DIAGNOSTIC MODE L      0= ENABLE  OUTPUT BITS 08-15,
                                ;          -------                                 1= DISABLE OUTPUT BITS 08-15
000400                          ODDP     := BIT08         ; ODD PARITY H  BIT07= 0 & 0= EVEN RAM PARITY, 1= ODD PARITY
                                ;        := BIT09         ; UNUSED HARDWARE BIT
002000                          DDC      := BIT10         ; DIAGNOSTIC DRIVE CLOCK
004000                          DDD      := BIT11         ; DIAGNOSTIC DRIVE DATA
                                ;          -------
010000                          LED1     := BIT12         ; LED 1 L, 0 = TURN LED ON, 1 = TURN LED OFF
020000                          LED2     := BIT13         ; LED 2 L       "                "
040000                          LED4     := BIT14         ; LED 4 L       "                "
100000                          LED8     := BIT15         ; LED 8 L       "                "
                                ;
170000                          LEDS     := LED8+LED4+LED2+LED1
                                ;
```



```
                                ;
                                ;                        SDI INTERCONNECT SIGNALS
                                ;
                                ; ------------+                                              +-------------
                                ;             !-----------WRITE/COMMAND DATA    (WCD)------------!
                                ; CONTROLLER  !-----REAL TIME CONTROLLER STATE FRAME (RTCS)---->!     DRIVE
                                ;             !<----------READ/RESPONSE DATA    (RRD)------------!
                                ;             !<----REAL TIME DRIVE STATE FRAME       (RTDS)-----!
                                ; ------------+                                              +-------------
                                ;
                                ;
                                ;BIT 15 ! 0   1   2   3 ! 4   5   6   7 ! 8   9  10  11 !12  13  14  15 ! 0
                                ;   -----+---------PREAMBLE--------------SYNC--------STATUS--------------+-----
                                ;    ! P ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 1 !RDY!WRT!RD !INI!RES!RES!PTY! 0 !
                                ;   -----+-------------------------------------------------------------+-----
                                ;        +                !                !                !N/T N/T         +
                                ;
                                ;
                                ;RTDS                     REAL TIME DRIVE STATE FRAME
                                ;
                                ;BIT 15 ! 0   1   2   3 ! 4   5   6   7 ! 8   9  10  11 !12  13  14  15 ! 0
                                ;   -----+---------PREAMBLE--------------SYNC--------STATUS--------------+-----
                                ;    ! P ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 0 ! 1 !RDY!ATN!R/W!SEC!IDX!AVL!PTY! 0 !
                                ;   -----+-------------------------------------------------------------+-----
                                ;
                                ; 1. EVEN PARITY OVER COMPLETE FRAME
                                ; 2. DATA SENT LEAST SIGNIFICANT BIT FIRST
                                ; 3. IN DIAGNOSTIC MODE IT TAKES 21 CLOCKS FROM LOADING RTCS UNTIL RTDS IS VALID
                                ; 4. N/T= DESERIALIZER RECEIVES NON TRI-STATED AND TRI-STATED.
                                ;
```

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;RTCS    := 05          ; REAL TIME CONTROLLER STATE                    -      WRITE
                                ;
                                ;       1.DO NOT CHANGE READ AND WRITE GATE IN THE SAME INSTRUCTION SINCE IT
                                ;         WOULD BE POSSIBLE FOR BOTH TO BE SENT ACTIVE.
                                ;       2.EACH TIME THIS REGISTER IS LOADED, ONE FRAME WILL BE SENT REGARDLESS
                                ;         OF THE STATE OF THE 'CONT H' BIT.
                                ;       3.BEFORE CHANGING PORTS, TURN OFF THE 'CONT H' BIT AND WAIT FOR 'RTCS
                                ;         READY' OF THE 'RTDS' REGISTER TO INDICATE THE TRANSMISSION HAS COMPLETED.
                                ;
000001                          CONT     := BIT00       ; +CONT H        0= SEND ZERO'S , 1= SEND FRAME
000002                          WRT      := BIT01       ; +WRT GATE H    0= NO WRITE GATE,1= ENABLE WRITTING OF DATA
000004                          RD       := BIT02       ; +RD GATE H     0= NO READ DATA, 1= SEND READ DATA
000010                          CRDY     := BIT03       ; +CON RCVR RDY H 0= NOT READY, 1= READY TO RECEIVE COMMAND/DATA
                                ;          -------
000020                          INI      := BIT04       ; +INIT H        0= NO INIT, 1= INITIALIZE (RESET) DRIVE
000040                          rtcs05   := bit05 ;temp ; +unused 1
000040                          RTCS13   := BIT05       ; +UNUSED 1      0= DEFAULT
000100                          rtcs06   := bit06 ;temp ; +unused 0
000100                          RTCS14   := BIT06       ; +UNUSED 0      0= DEFAULT
000200                          FPE      := BIT07       ; +FORCE PARITY ERROR H 0= SEND (CORRECT) EVEN PARITY,1= SEND ODD PARITY
                                ;          -------
                                ;        := BIT08       ; +RESERVED
                                ;        := BIT09       ; +      "
                                ;        := BIT10       ; +      "
                                ;        := BIT11       ; +      "
                                ;          --------
                                ;        := BIT12       ; +      "
                                ;        := BIT13       ; +      "
                                ;        := BIT14       ; +      "
                                ;        := BIT15       ; +RESERVED
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;RTDS    := 02          ; REAL TIME DRIVE STATE (NON EXISTANT)          -      READ
                                ;
                                ;       WHEN FIRST SELECTING A PORT, THE FIRST 'RTDS' FRAME MUST BE DISREGARDED.
                                ;       TO DO THIS, SEND ONLY ONE 'RTCS' FRAME THEN WAIT FOR 'FRAME SENT' TO
                                ;       GO ACTIVE
                                ;
000001                          DRDY     := BIT00       ; +D RCVR RDY H 0= DO NOT SEND,  1= RECEIVER READY FOR COMMAND
000002                          ATTN     := BIT01       ; +ATTN H        0= NO ERROR,    1=
                                ;        := BIT02       ; +RESERVED
                                ;        := BIT03       ; +      "
                                ;          --------
000020                          SEC      := BIT04       ; +SECTOR H      0= NO SECTOR    1= SECTOR PULSE(USE 1-0 EDGE)
000040                          IDX      := BIT05       ; +INDEX H       0= NO INDEX     1= INDEX PULSE (USE 1-0 EDGE)
000100                          rtds06   := bit06 ;temp ;
000100                          AVAIL    := BIT06       ; +AVAILABLE     0= NOT AVAILABLE 1= DEVICE AVAILABLE FOR USE
                                ;        := BIT07       ; +RESERVED
                                ;          --------
                                ;        := BIT08       ; +      "
                                ;        := BIT09       ; +      "
                                ;        := BIT10       ; +      "
                                ;        := BIT11       ; +      "
                                ;          --------
                                ;        := BIT12       ; +      "
                                ;        := BIT13       ; +      "
                                ;        := BIT14       ; +      "
100000                          RWRDY    := BIT15       ; +R/W RDY H     0= NOT R/W READY, 1= READ/WRITE READY
```

```
;MNEMONIC  ADDRESS      DESCRIPTION                                   UPROC   DPROC
;
;PAR     := 02          ; PROGRAM ADDRESS REGISTER                    WRITE   WRITE
;
;               A  WRITE  TO THIS REGISTER WILL CAUSE A JUMP TO AN ADDRESS
;               BETWEEN FOOH AND FFF HEXIDECIMAL. THE INSTRUCTION JUMPED TO MUST
;               CONTAIN A  JUMP  INSTRUCTION TO PREVENT A FALL BACK TO THE
;               INSTRUCTION AFTER THE PAR UPDATE.
;
;                  15 +  NOT  IMPLEMENTED
;                  14 +   '       '
;                  13 +   '       '
;                  12 +   '       '
;                --------
;                  11 +MEMORY ADDRESS 11 = ALWAYS ONE
;                  10 +   '       '   10 = ALWAYS ONE
;                  09 +   '       '   09 = ALWAYS ONE
;                  08 +   '       '   08 = ALWAYS ONE
;                --------
;                  07 +   '       '   07  (  128)
;                  06 +   '       '   06  (   64)
;                  05 +   '       '   05  (   32)
;                  04 +   '       '   04  (   16)
;                --------
;                  03 +   '       '   03  (    8)
;                  02 +   '       '   02  (    4)
;                  01 +   '       '   01  (    2)
;                  00 +   '       '   00  (    1)
```



```
;MNEMONIC  ADDRESS      DESCRIPTION                                   UPROC   DPROC
;
;BUF     := 03          ; RAM DATA BUFFER                             R/W     R/W
;
;       DATA TO OR FROM INTERNAL REGISTERS MUST BE MOVED WITH A LOGICAL
;       OPERATION.
;
;            15 +BUFFER  DATA   15  (32768)
;            14 +   '       '   14  (16384)
;            13 +   '       '   13  ( 8196)
;            12 +   '       '   12  ( 4096)
;          --------
;            11 +   '       '   11  ( 2048)
;            10 +   '       '   10  ( 1024)
;            09 +   '       '   09  (  512)
;            08 +   '       '   08  (  256)
;          --------
;            07 +   '       '   07  (  128)
;            06 +   '       '   06  (   64)
;            05 +   '       '   05  (   32)
;            04 +   '       '   04  (   16)
;          --------
;            03 +   '       '   03  (    8)
;            02 +   '       '   02  (    4)
;            01 +   '       '   01  (    2)
;            00 +   '       '   00  (    1)
```

```
                                                                        UPROC   DPROC
;UDS     := 15    ; BUS ADDR/DATA REGISTER SOURCE                       READ    (RD)
;UDD     := 05    ; BUS ADDR/DATA REGISTER DESTINATION                  WRITE   (RD)
;
;       SEE I/O REGISTER FOR FORMAT DURING INITIALIZATION
;
;       DMA TRANSFERS - DMA REQUEST @SDMA MUST BE MADE AT LEAST
;       ONE INSTRUCTION BEFORE READING OR WRITING FROM THE BUS DATA REG.
;       ONCE BUS MASTER HAS BEEN GRANTED, DMA REQUEST MAY BE DROPPED @RDMA.
;
;       THE FOLLOWING BIT DEFINITIONS APPLY TO THIS REGISTER BEING USED AS A
;       BUS ADDRESS REGISTER.
;
;       LO BUS ADDRESS BITS 0-15
;
;            15 +  BUS ADDRESS  15  (32768)
;            14 +   '       '   14  (16384)
;            13 +   '       '   13  ( 8196)
;            12 +   '       '   12  ( 4096)
;          --------
;            11 +   '       '   11  ( 2048)
;            10 +   '       '   10  ( 1024)
;            09 +   '       '   09  (  512)
;            08 +   '       '   08  (  256)
;          --------
;            07 +   '       '   07  (  128)
;            06 +   '       '   06  (   64)
;            05 +   '       '   05  (   32)
;            04 +   '       '   04  (   16)
;          --------
;            03 +   '       '   03  (    8)
;            02 +   '       '   02  (    4)
;            01 +   '       '   01  (    2)
;            00 +   '       '   00  (    1)
;
```



```
;
;       THE FOLLOWING BIT DEFINITIONS APPLY TO THIS REGISTER BEING USED AS A
;       DATA REGISTER.
;
;            15 +  BUS   DATA   15  (32768)
;            14 +   '       '   14  (16384)
;            13 +   '       '   13  ( 8196)
;            12 +   '       '   12  ( 4096)
;          --------
;            11 +   '       '   11  ( 2048)
;            10 +   '       '   10  ( 1024)
;            09 +   '       '   09  (  512)
;            08 +   '       '   08  (  256)
;          --------
;            07 +   '       '   07  (  128)
;            06 +   '       '   06  (   64)
;            05 +   '       '   05  (   32)
;            04 +   '       '   04  (   16)
;          --------
;            03 +   '       '   03  (    8)
;            02 +   '       '   02  (    4)
;            01 +   '       '   01  (    2)
;            00 +   '       '   00  (    1)
;
```

```
                                ;                                                                   UPROC   DPROC
                                ;BREG    := 06    ; BI REGISTER FOR BI CONTROL                      WRITE    -
                                ;
000001                          B.NABO   := BIT00                 ; ABORT BI COMMAND=0, NOT ABORT=1
                                ;        := BIT01                 ; NOT USED
                                ;        := BIT02                 ; NOT USED
000010                          B.BAD    := BIT03                 ; NODE NOT BAD=0, NODE BAD=1
                                ;          --------
000020                          B.LED    := BIT04                 ; YELLOW LED OFF=0, YELLOW LED ON=1
000040                          B.ODD    := BIT05                 ; BYTE BOUNDARIES IN BCAI; EVEN=0, ODD=1
000100                          B.LOOP   := BIT06                 ; BI BUS REQUEST; NORMAL MODE=0, LOOPBACK MODE=1
000200                          B.NRTY   := BIT07                 ; RETRY COMMAND; RETRY=0, NO RETRY=1
                                ;           --------
                                ;        := BIT08                 ; NOT USED
                                ;        := BIT09                 ; NOT USED
                                ;        := BIT10                 ; NOT USED
                                ;        := BIT11                 ; NOT USED
                                ;         --------
                                ;        := BIT12                 ; NOT USED
                                ;        := BIT13                 ; NOT USED
                                ;        := BIT14                 ; NOT USED
                                ;        := BIT15                 ; NOT USED
                                ;
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                  UPROC   DPROC
                                ;
                                ;SD      := 06           ; SERDES 16 DATA REGISTER                    -      R/W
                                ;
                                ;       -+--------HEADER---------+----------+--------SS-------------+---SS---+-
                                ;        !LBN!LBN!LBN!LBN!LBN!LBN!          !SYN!    DATA       !EDC!  ECC   !
                                ;       -+-----------------------+----------+--------SS-------------+---SS---+-
                                ;
                                ;       SW1 = SOFTWARE WORD 1
                                ;       SW2 = SOFTWARE WORD 2
                                ;       EDC = ERROR DETECTION CODE
                                ;       ECC = ERROR CORRECTION CODE
                                ;       SYN = 23274Q, 26BCH, 9916
                                ;
                                ;            15 +SERDES  DATA   15  (32768)
                                ;            14 +    '       '  14  (16384)
                                ;            13 +    '       '  13  ( 8196)
                                ;            12 +    '       '  12  ( 4096)
                                ;          --------
                                ;            11 +    '       '  11  ( 2048)
                                ;            10 +    '       '  10  ( 1024)
                                ;            09 +    '       '  09  (  512)
                                ;            08 +    '       '  08  (  256)
                                ;          --------
                                ;            07 +    '       '  07  (  128)
                                ;            06 +    '       '  06  (   64)
                                ;            05 +    '       '  05  (   32)
                                ;            04 +    '       '  04  (   16)
                                ;          --------
                                ;            03 +    '       '  03  (    8)
                                ;            02 +    '       '  02  (    4)
                                ;            01 +    '       '  01  (    2)
                                ;            00 +    '       '  00  (    1)
```


LSCS FORM=QUAD

```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;ECC      := 07         ; RS GENERATOR REGISTER (RESIDUE)              -      READ
                                ;
                                ;         := BIT15      ; 0= ECC ERROR, 1= NO ECC ERROR
                                ;         := BIT14      ; +ECC FEEDBACK ENABLE
                                ;         := BIT13      ; -WRITE ECC=NOT (ECC ENABLE AND ECC INPUT ENABLE)
                                ;         := BIT12      ; -DATA OUT
                                ;           --------
                                ;         := BIT11      ; +SERDES ENABLE
                                ;         := BIT10      ; +DTEST                                  @IOC.D
                                ;         := BIT09      ; +  RS GENERATOR 09  (  512)
                                ;         := BIT08      ; +    '        '    08  (  256)
                                ;           --------
                                ;         := BIT07      ; +    '        '    07  (  128)
                                ;         := BIT06      ; +    '        '    06  (   64)
                                ;         := BIT05      ; +    '        '    05  (   32)
                                ;         := BIT04      ; +    '        '    04  (   16)
                                ;           --------
                                ;         := BIT03      ; +    '        '    03  (    8)
                                ;         := BIT02      ; +    '        '    02  (    4)
                                ;         := BIT01      ; +    '        '    01  (    2)
                                ;         := BIT00      ; +    '        '    00  (    1)
```



```
                                ;MNEMONIC  ADDRESS      DESCRIPTION                                    UPROC   DPROC
                                ;
                                ;BAR      := 07         ; BUFFER ADDRESS REGISTER  (BUS PROC)        WRITE     -
                                ;BAR      := 07         ; BUFFER ADDRESS REGISTER  (DRIVE PROC)        -      WRITE
                                ;
                                ;         THE LOWER 4 BITS ARE INCREMENTED EVERY TIME THE DATA BUFFER IS WRITTEN
                                ;         INTO. THEY ARE NOT INCREMENTED WHEN THE DATA BUFFER IS READ.
                                ;
                                ;              15       ; NOT IMPLEMENTED
                                ;              14       ;  "       "
                                ;              13       ; BUFFER ADDRESS 13   ( 8192)
                                ;              12       ;  "       "     12   ( 4096)
                                ;            --------
                                ;              11       ;  "       "     11   ( 2048)
                                ;              10       ;  "       "     10   ( 1024)
                                ;              09       ;  "       "     09   (  512)
                                ;              08       ;  "       "     08   (  256)
                                ;            --------
                                ;              07       ;  "       "     07   (  128)
                                ;              06       ;  "       "     06   (   64)
                                ;              05       ;  "       "     05   (   32)
                                ;              04       ;  "       "     04   (   16)
                                ;            --------
                                ;              03       ;  "       "     03   (    8)
                                ;              02       ;  "       "     02   (    4)
                                ;              01       ;  "       "     01   (    2)
                                ;              00       ;  "       "     00   (    1)
```

```
                        ;       *-------------------------------------------------------------------*
                        ;       *                        BUS  I/O REGISTERS                         *
                        ;       *-------------------------------------------------------------------*
                        ;
                        ;   REFERENCE "BUS DISK ADAPTER FUNCTIONAL SPECIFICATION" SECTION 4.3.4.3
                        ;
                        ;       TO SUMMARIZE THE BUS I/O REGISTER ADDRESSES AND INTERRUPTS:
                        ;
                        ;xDAIP  : 7XXXXOR.INIT= UNDEFINED                XXO =   MESSAGE RECEIVED
                        ;xDAIP  :       OR.RUN = POLL I/O BUFFER                 ADAPTOR READY
                        ;xDAIP  :       OW.R/I = CONTROLLER INIT                 REQUEST VAX PURGE
                        ;xDASA  : 7XXXX2W.INIT= INITIALIZATION STATUS
                        ;xDASA  : 7XXXX2R.INIT= INITIALIZATION STATUS
                        ;xDASA  : 7XXXX2R.RUN = RESERVED
                        ;xDASA  : 7XXXX2W.RUN = VAX PURGE COMPLETE (HOST RESPONSE)
                        ;
                        ;       1. ANYTIME A xDA BUS I/O PAGE ADDRESS IS SELECTED, A SLAVE SYNC
                        ;          (SSYNC L) IS RETURNED.
                        ;       2. THE xDA HAS TWO (2) MODES OF OPERATION:
                        ;             1. INITIALIZATION MODE - THE xDA IS RUNNING DIAGNOSTICS AND
                        ;                GETTING INFORMATION TO BE USED IN RUN MODE.
                        ;             2. RUN MODE - THE xDA WILL ACCEPT AND EXECUTE DCP PACKETS.
                        ;       3. INFORMATION SENT OR RECEIVED FROM THE xDA I/O PAGE IS VALID ONLY
                        ;          WHEN IN THE SPECIFIED MODE.
                        ;
                        ; *-------------------------------------------------------------------------*
                        ; *     BUS HARDWARE USAGE                                                  *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ;                              +-----------------------------------+
                        ;                       ON=  1 !  1  2  3  4  5  6  7  8  9 10  !
                        ;        +A2 -A2               !        BUS ADDRESS SWITCH       !
                        ;       0---0---0      OFF= 0 !  4  8 16 32 64 128 256 512 1K 2K  !
                        ;         W5  W4               +-----------------------------------+
                        ;
```



```
                        ;
                        ; *-------------------------------------------------------------------------*
                        ; *     xDASA INITIALIZATION REGISTER BIT DEFINITIONS                       *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ; *-------------------------------------------------------------------------*
                        ; *     xDA WRITE FORMAT                                                    *
                        ; *-------------------------------------------------------------------------*
                        ;
                        ;xDASA  1W.INIT  STEP 1-3 WRITE FORMAT
                        ;
                        ;         --------
                        ;        := BIT00        ; +ERROR CODE BIT 0
                        ;        := BIT01        ; +ERROR CODE BIT 1
                        ;        := BIT02        ; +ERROR CODE BIT 2
                        ;        := BIT03        ; +ERROR CODE BIT 3
                        ;         --------
                        ;        := BIT04        ; +ERROR CODE BIT 4
                        ;        := BIT05        ; +ERROR CODE BIT 5
                        ;        := BIT06        ; +ERROR CODE BIT 6
                        ;        := BIT07        ; +ERROR CODE BIT 7
                        ;         --------
004000                  STEP1   := BIT11        ; INIT STEP 1 INDICATOR
010000                  STEP2   := BIT12        ; INIT STEP 2 INDICATOR
020000                  STEP3   := BIT13        ; INIT STEP 3 INDICATOR
040000                  STEP4   := BIT14        ; INIT STEP 4 INDICATOR
100000                  ERR     := BIT15        ; +INITIALIZATION ERROR FLAG
                        ;         --------
000200                  INTI    := BIT07        ; INIT INTERRUPT ENABLE BIT (ECHOED TO HOST)
000200                  IE      := BIT07        ; INIT INTERRUPT ENABLE BIT (ECHOED TO HOST)
100000                  OWN     := BIT15        ; 0 = HOST OWNS RING ENTRY, 1 = xDA OWNS ENTRY
040000                  FLAG    := BIT14        ; 0 = NO INTERRUPTS, 1 = INTERRUPT IF APPROPRIATE
                        ;         --------
```

```
                                ;
                                ; *-------------------------------------------------------------------------------*
                                ; *      xDA READ FORMATS                                                         *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ; *-------------------------------------------------------------------------------*
                                ; *      STEP 1 FORMAT                                                            *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ;XXXX    1R.INIT  STEP 1 READ FORMAT
                                ;   BITS 15 - 0 ; +LO ORDER 16 BITS OF CSR BASE ADDRESS
                                ;
                                ; *-------------------------------------------------------------------------------*
                                ; *      STEP 2 FORMAT                                                            *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ;XXXX    1/3R.INIT  STEP 2 READ FORMAT
                                ;
                                ; *-------------------------------------------------------------------------------*
                                ; *      STEP 3 FORMAT                                                            *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ;XXXX    1R.INIT  STEP 3 READ FORMAT
                                ;
                                ;   BITS 15 - 1                     ; RESERVED
000001                          GO      := BIT00                    ; +GO BIT (SET BY HOST TO PUT xDA ONLINE)
                                ;
000265                          INSTR   := <165.*110.>/100.         ;NSEC U-INSTRUCTION CYCLE TIME + DEVIATION
000552                          CYCLE   := <INSTR*2>                ;NSEC U-INSTRUCTION CYCLE TIME NOMINAL (PROCS)
```



```
                                ;
                                ; *-------------------------------------------------------------------------------*
                                ; *                        LED CODE DEFINITION                                    *
                                ; *-------------------------------------------------------------------------------*
                                ;
                                ;                        LED LED LED LED
                                ;                         8   4   2   1
                                ;                        --- --- --- ---
                                ;                    0 =  0   0   0   0   NO ERROR (OR STUCK IN DIAGNOSTIC IF ALWAYS OFF)
                                ;                    1 =  0   0   0   1   NOT USED
                                ;                    2 =  0   0   1   0   INITIALIZATION STEP 2
                                ;                    3 =  0   0   1   1   INITIALIZATION STEP 3
                                ;                    4 =  0   1   0   0   INITIALIZATION STEP 4
                                ;                    5 =  0   1   0   1   1 SECOND BLINK FOR BUSY- NO BLINK FOR HANG
                                ;                    6 =  0   1   1   0   NOT USED
                                ;                    7 =  0   1   1   1   NOT USED
                                ;
                                ;                        ERROR CODES
                                ;                        -----------
                                ;                    8 =  1   0   0   0   INITIALIZATION (ERROR)/STEP 1 DIAGNOSTIC WRAP
                                ;                    9 =  1   0   0   1   ERROR= DIAGNOSTIC ERROR xDA #1
                                ;                    A =  1   0   1   0   ERROR= DIAGNOSTIC ERROR xDA #2
                                ;                    B =  1   0   1   1   ERROR= NOT USED
                                ;                    C =  1   1   0   0   ERROR= ERROR VECTOR = ROM  PARITY ERROR
                                ;                    D =  1   1   0   1   ERROR= ERROR VECTOR = RAM  PARITY ERROR
                                ;                    E =  1   1   1   0   ERROR= ERROR VECTOR = ROM/RAM PARITY ERROR
                                ;                    F =  1   1   1   1   ERROR= SEQUENCER ERROR
                                ;
000017                          ERREG   := R17
                                ;
                                ; ***   LED4 ON INDICATES THE xDA IS OPERATIONAL ***
                                ;
                                ;;xDAOK := LED4                          ; BIC CODE FOR xDA OK LED DISPLAY
                                ;
                                ; ***   LED1 ON INDICATES THAT THE D.PROC IS BUSY, OFF MEANS IT IS IDLE ***
                                ;
                                ;;DBUSY := LED1                          ; CODE FOR D.PROC BUSY LED DISPLAY
                                ;
                                ; ***   LED2 ON INDICATES THAT THE U.PROC IS BUSY, OFF MEANS IT IS IDLE ***
                                ;
                                ;;UBUSY := LED2                          ; CODE FOR U.PROC BUSY LED DISPLAY
```

```
                                ;*------------------------------------------------------------------------------*
                                ;*                  xDA ERROR CODES                                             *
                                ;*------------------------------------------------------------------------------*
                                ;
                                ;       THE ERROR DISPLAY IS BIT ENCODED TO CORRESPOND TO THE ERROR MESSAGE
                                ;
                                ;               BIT 15  0 = ERROR, 1 = NO ERROR
                                ;               BIT 14  ERROR STATUS 7 BIT
                                ;               BIT 13  ERROR STATUS 6 BIT
                                ;               BIT 12  ERROR STATUS 5 BIT
                                ;
                                ;                       SELF TEST ERROR CODES
                                ;
104000                          ERR00   := <ERR+STEP1>  ;ERROR IN STACK TEST
000040                          ERR01   := 1 *32.       ;ERROR IN ALU OPERATIONAL TEST/CONTROL PROM PE/ALG&LOG PROM/CR TEST
000100                          ERR02   := 2 *32.       ;ERROR IN RAM PE/RAM BUFFER TEST
000140                          ERR03   := 3 *32.       ;ERROR in SDI/SERDES/ECC TEST
000200                          ERR04   := 4 *32.       ;ERROR in STEP 1 INIT ERROR
000240                          ERR05   := 5 *32.       ;ERROR in NPR R/W TEST
000300                          ERR06   := 6 *32.       ;ERROR in STEP 2,3, OR 4 INIT ERROR
000340                          ERR07   := 7 *32.       ;ERROR in Q BUS MEMORY TEST (MANUFACTURING ONLY)
000340                          ERR07   := 7 *32.       ;ERROR in BI TEST ERRORS
```



```
                                ;*------------------------------------------------------------------------------*
                                ;*      HI WORD BIT DEFINTIONS FOR BDA                                          *
                                ;*------------------------------------------------------------------------------*

100000                          BIT31   := BIT15        ; BIT 15 OF HI WORD
040000                          BIT30   := BIT14        ; BIT 14        "
020000                          BIT29   := BIT13        ; BIT 13        "
010000                          BIT28   := BIT12        ; BIT 12        "
                                ;          -------
004000                          BIT27   := BIT11        ; BIT 11        "
002000                          BIT26   := BIT10        ; BIT 10        "
001000                          BIT25   := BIT09        ; BIT 09        "
000400                          BIT24   := BIT08        ; BIT 08        "
                                ;          -------
000200                          BIT23   := BIT07        ; BIT 07        "
000100                          BIT22   := BIT06        ; BIT 06        "
000040                          BIT21   := BIT05        ; BIT 05        "
000020                          BIT20   := BIT04        ; BIT 04        "
                                ;          --------
000010                          BIT19   := BIT03        ; BIT 03        "
000004                          BIT18   := BIT02        ; BIT 02        "
000002                          BIT17   := BIT01        ; BIT 01        "
000001                          BIT16   := BIT00        ; BIT 00        "
```

```
                                ; *-------------------------------------------------------------------------------*
                                ; *      BDA COMMAND CODES EQUATES                                                *
                                ; *-------------------------------------------------------------------------------*

000001                          BIRDCM   := 1
000001                          READ     := 1
000002                          IRCI     := 2
000003                          RCI      := 3
000004                          BIWRCM   := 4
000004                          WRITE    := 4
000005                          WCI      := 5
000006                          UWMCI    := 6
000007                          WMCI     := 7
000010                          INTR     := ^H08
000011                          IDENT    := ^H09
                                ;STOPC   := ^H0A
                                ;BISTCM  := ^H0B
000014                          INVAL    := ^H0C
000015                          BDCST    := ^H0D
000016                          IPINTR   := ^H0E
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC GENERAL REGISTER OFFSET EQUATES                                     *
                                ; *-------------------------------------------------------------------------------*

000000                          BIDTYP   := ^H000        ;OFFSET TO DEVICE REGISTER
000004                          BICSR    := ^H004        ;OFFSET TO CONTROL AND STATUS REGISTER
000010                          BIBER    := ^H008        ;OFFSET TO BUS ERROR REGISTER
000014                          BIECSR   := ^H00C        ;OFFSET TO ERROR INTERRUPT CONTROL REGISTER
000020                          BIIDES   := ^H010        ;OFFSET TO INTERRUPT DESTINATION REGISTER

                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC SPECIFIC DEVICE REGISTER OFFSET EQUATES                             *
                                ; *-------------------------------------------------------------------------------*

000024                          BIIMSK   := ^H014        ;OFFSET TO IPINTR MASK REGISTER
000030                          BIPSDE   := ^H018        ;OFFSET TO FORCE IPINTR/STOP DESTINATION REGISTER
000034                          BIPSRC   := ^H01C        ;OFFSET TO IPINTR SOURCE REGISTER
000040                          BISADR   := ^H020        ;OFFSET TO STARTING ADDRESS REGISTER
000044                          BIEADR   := ^H024        ;OFFSET TO ENDING ADDRESS REGISTER
000050                          BCICSR   := ^H028        ;OFFSET TO BCI CONTROL REGISTER
000054                          BIWSTA   := ^H02C        ;OFFSET TO WRITE STATUS REGISTER
000060                          BIPSFC   := ^H030        ;OFFSET TO FORCE IPINTR/STOP COMMAND REGISTER
000100                          BIUCSR   := ^H040        ;OFFSET TO USER INTERFACE INTERRUPT CONTROL REGISTER
000360                          BIGPR0   := ^HOF0        ;OFFSET TO GENERAL PURPOSE REGISTER 0
000364                          BIGPR1   := ^HOF4        ;OFFSET TO GENERAL PURPOSE REGISTER 1
000370                          BIGPR2   := ^HOF8        ;OFFSET TO GENERAL PURPOSE REGISTER 2
000374                          BIGPR3   := ^HOFC        ;OFFSET TO GENERAL PURPOSE REGISTER 3

                                ; *-------------------------------------------------------------------------------*
                                ; *      BIIC UQSSP REGISTER OFFSET EQUATES                                       *
                                ; *-------------------------------------------------------------------------------*

000362                          IPREG    := ^HOF2        ;OFFSET TO IP REGISTER (ODD WORD BOUNDRY OF GPR 0)
000364                          SAREG    := ^HOF4        ;OFFSET TO SA REGISTER (EVEN WORD BOUNDRY OF GPR 1)
```

```
                        ; *-------------------------------------------------------------------------*
                        ; * BIDTYP      ;^HOOO  ;BIIC DEVICE REGISTER                               *
                        ; *-------------------------------------------------------------------------*

000416                  B.DTYP  := ^H10E        ;DEVICE TYPE

                        ; *-------------------------------------------------------------------------*
                        ; * BICSR       ;^HOO4  ;BIIC CONTROL AND STATUS REGISTER EQUATES           *
                        ; *-------------------------------------------------------------------------*

100000                  HES     := BIT15        ;HARD ERROR SUMMARY
040000                  SES     := BIT14        ;SOFT ERROR SUMMARY
010000                  BROKE   := BIT12        ;BROKE BIT
004000                  STS     := BIT11        ;SELF-TEST STATUS BIT
002000                  SST     := BIT10        ;START SELF-TEST BIT

000017                  NODEID  := 17           ;NODE ID MASK

                        ; *-------------------------------------------------------------------------*
                        ; * BIBER       ;^HOO8  ;BUS ERROR REGISTER                                 *
                        ; *-------------------------------------------------------------------------*

040000                  B.NMR   := BIT30        ;NO ACK TO MULTI-RESPONDER CMD RECV
020000                  B.MTCE  := BIT29        ;MASTER TRANSMIT CHECK ERROR
010000                  B.CTE   := BIT28        ;CONTROL TRANSMIT ERROR
004000                  B.MPE   := BIT27        ;MASTER PARITY ERROR
002000                  B.ISE   := BIT26        ;INTERLOCK SEQUENCE ERROR
001000                  B.TDF   := BIT25        ;TRANSMITTER DURING FAULT
000400                  B.IVE   := BIT24        ;IDENT VECTOR ERROR
000200                  B.CPE   := BIT23        ;COMMAND PARITY ERROR
000100                  B.SPE   := BIT22        ;SLAVE PARITY ERROR
000040                  B.RDS   := BIT21        ;READ DATA SUBSTITUTE
000020                  B.RTO   := BIT20        ;RETRY TIMEOUT
000010                  B.STO   := BIT19        ;STALL TIMEOUT
000004                  B.BTO   := BIT18        ;BUS TIMEOUT
000002                  B.NEX   := BIT17        ;NON-EXISTENT ADDRESS
000001                  B.ICE   := BIT16        ;ILLEGAL CONFIRMATION ERROR

                        ; *-------------------------------------------------------------------------*
                        ; * BISADR      ;^HO20  ;STARTING ADDRESS REGISTER                          *
                        ; * BIEADR      ;^HO24  ;ENDING ADDRESS REGISTER                            *
                        ; *-------------------------------------------------------------------------*

020000                  BA29    := BIT29        ; BUS ADDRESS 29
010000                  BA28    := BIT28        ; BUS ADDRESS 28
004000                  BA27    := BIT27        ; BUS ADDRESS 27
002000                  BA26    := BIT26        ; BUS ADDRESS 26
001000                  BA25    := BIT25        ; BUS ADDRESS 25
000400                  BA24    := BIT24        ; BUS ADDRESS 24
000200                  BA23    := BIT23        ; BUS ADDRESS 23
000100                  BA22    := BIT22        ; BUS ADDRESS 22
000040                  BA21    := BIT21        ; BUS ADDRESS 21
000020                  BA20    := BIT20        ; BUS ADDRESS 20
000010                  BA19    := BIT19        ; BUS ADDRESS 19
```



```
000004                  BA18    := BIT18        ; BUS ADDRESS 18
000002                  BA17    := BIT17        ; BUS ADDRESS 17
000001                  BA16    := BIT16        ; BUS ADDRESS 16

037777                  BA      := BA16+BA17+BA18+BA19+BA20+BA21+BA22+BA23+BA24+BA25+BA26+BA27+BA28+BA29

                        ; *-------------------------------------------------------------------------*
                        ; * BCICSR      ;^HO28  ;BCI CONTROL REGISTER EQUATES                       *
                        ; *-------------------------------------------------------------------------*

000001                  FORCE   := BIT16        ;FORCE COMMAND

020000                  STOPEN  := BIT13        ;STOP ENABLE
000400                  UCSREN  := BIT08        ;USER CSR SPACE ENABLE
000200                  BICSREN := BIT07        ;BIIC CSR SPACE ENABLE
000010                  RTOEVEN := BIT03        ;RTO EV CODE ENABLE

                        ; *-------------------------------------------------------------------------*
                        ; * BIWSTA      ;^HO2C  ;WRITE STATUS REGISTER                              *
                        ; *-------------------------------------------------------------------------*

100000                  GPR3    := BIT31        ;GENERAL PURPOSE REGISTER 3 STATUS BIT
040000                  GPR2    := BIT30        ;GENERAL PURPOSE REGISTER 2 STATUS BIT
020000                  GPR1    := BIT29        ;GENERAL PURPOSE REGISTER 1 STATUS BIT
010000                  GPRO    := BIT28        ;GENERAL PURPOSE REGISTER O STATUS BIT

                        ; *-------------------------------------------------------------------------*
                        ; * BIPSFC      ;^HO30  ;FORCE IPINTR/STOP COMMAND REGISTER EQUATES         *
                        ; *-------------------------------------------------------------------------*

140000                  STOPCM  := BIT15+BIT14  ;STOP COMMAND

                        ; *-------------------------------------------------------------------------*
                        ; * BIUCSR      ;^HO40  ;USER INTERFACE INTERRUPT CONTROL REGISTER EQUATES  *
                        ; *-------------------------------------------------------------------------*

000001                  BRLV4   :=      BIT16   ; FORCE BITS 16-19
000002                  BRLV5   :=      BIT17
000004                  BRLV6   :=      BIT18
000010                  BRLV7   :=      BIT19

000020                  BRLV4S  :=      BIT20   ; INTRC BITS 20-23
000040                  BRLV5S  :=      BIT21
000100                  BRLV6S  :=      BIT22
000200                  BRLV7S  :=      BIT23

000400                  BRLV4R  :=      BIT24   ; INTRC BITS 24-27
001000                  BRLV5R  :=      BIT25
002000                  BRLV6R  :=      BIT26
004000                  BRLV7R  :=      BIT27
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *      BI COMMAND CODES                                                   *
                        ; *-------------------------------------------------------------------------*
                        ;
140000                  QW      := BIT31+BIT30  ;OCTAWORD DATA LENGTH
100000                  QW      := BIT31        ;QUADWORD DATA LENGTH
040000                  LW      := BIT30        ;LONGWORD DATA LENGTH
000000                  WD      := 0            ;WORD DATA LENGTH (only works for BI registers)
020000                  IOACC   := BIT29        ;IO SPACE ACCESS
```



```
                        .SBTTL  CONTROLLER RAM AND EQUATE DEFINITIONS                                   ;[E121]

                        ; *-------------------------------------------------------------------------*
                        ; *      EQUATES FOR MSCP,SDI,BIT AND BYTE MASKS                            *
                        ; *-------------------------------------------------------------------------*

000100                  ATTCOD  := 100                  ; MASK FOR ATTENTION OP CODES
004000                  BTROTH  := BIT11                ; BETROTHED BIT IN GET STUD STATUS RESPONSE
000041                  BUFLMT  := 33.                  ; BUFFER CONTROL BLOCK ALLOCATION LIMIT
000400                  CCLASS  := 1*256.               ; MASS STOR. CONTROLLER CLASS FIELD FOR MSCP     [rae04]
000022                  CMODEL  := 18.                  ; CONTROLLER MODEL FIELD FOR MSCP
000024                  CMDLIM  := 20.                  ; MAXIMUM NUMBER OF MSCP PACKETS
170000                  CYLSTR  := 170000               ; MASK FOR STARTING HI CYLINDER
000377                  DEVCL   := 377                  ; MASK FOR MSCP DEVICE CLASS FIELD
001000                  DCLASS  := 1000                 ; DISK CLASS
000002                  DMBPC   := 2                    ; DM ODT PC SAVE AREA OFFSET FROM UN.BUF
000012                  DMODT   := 10.                  ; DM ODT LOCATION OFFSET FROM UN.BUF(START OF DM SPACE)
000011                  DMOVH   := 9.                   ; DM OVERLAY HI ADDR OFFSET FROM UN.BUF
000010                  DMOVL   := 8.                   ; DM OVERLAY LO ADDR OFFSET FROM UN.BUF
000013                  DMSTR   := 11.                  ; DM START ADDRESS OFFSET FROM UN.BUF
000007                  DSTSH   := 7                    ; DUST STATUS HI OFFSET FROM UN.BUF
000006                  DSTSL   := 6                    ; DUST STATUS LO OFFSET FROM UN.BUF
001000                  DUPVC   := 1000                 ; VIRTUAL CIRCUIT FOR DUP PROTOCOL
000105                  EDSEED  := 69.                  ; EDC SEED
000004                  ERRRTC  := 4                    ; ERROR RETRY COUNT
000000                  FCT.MD  := 0                    ; FCT MEDIA FORMAT WORD OFFSET
000002                  FCT.V1  := 2                    ; FCT VOLUME ID WORD 1 OFFSET
000003                  FCT.V2  := FCT.V1+1             ; FCT VOLUME ID WORD 2 OFFSET
177400                  FRMCD   := 177400               ; MASK FOR SDI FRAME CODE
170000                  HDCOD   := 170000               ; MASK FOR SDI HEADER CODE
001000                  HDRTMO  := 512.                 ; HEADER TIMEOUT VALUE - EQUALS ABOUT 170 US.
177400                  HIBYT   := 177400               ; MASK FOR HI BYTE OF WORD
000360                  HINIB   := 360                  ; MASK FOR HI NIBBLE
000377                  LOBYT   := 377                  ; MASK FOR LO BYTE OF WORD
000377                  HOSTF   := LOBYT                ; MASK FOR HOST UNIT FLAGS
000060                  IOMSK   := 60                   ; MASK FOR MSCP I/O COMMANDS
000377                  LBNMSK  := LOBYT                ; MASK FOR SDI LBN
007400                  LBNSTR  := 7400                 ; MASK FOR STARTING HI LBN
000100                  LOGCOD  := 100                  ; MASK FOR LOG/ATTENTION OP CODES
000017                  LONIB   := 17                   ; MASK FOR LO NIBBLE
100000                  MAPENB  := BIT15                ; MAPPING ENABLED BIT IN BUFF DESC
000003                  MAPMSK  := BIT00+BIT01          ; MASK FOR MAPPING REGISTER LOW 2 BITS
077700                  MAPMSH  := 77700                ; MASK FOR UNUSED HIGH BITS
000077                  MAPMSC  := 77                   ; COMPLEMENT OF MAPMSH
100000                  MAPVAL  := BIT15                ; MAPPING VALID BIT IN PTE
126736                  MD512   := 126736               ; FCT 512 FORMAT CODE
000042                  MEMSZ   := 34.                  ; NUMBER OF PTE'S THAT WE CAN CACHE
000074                  MINUTE  := 60.                  ; 60 SECONDS IN A MINUTE !
000003                  ODTAF   := 3                    ; DM ODT ARITMETIC FLAGS OFFSET FROM UN.BUF
000005                  ODTCA   := 5                    ; DM ODT CONSOLE ADDRESS OFFSET FROM UN.BUF
000004                  ODTCY   := 4                    ; DM ODT CARRY FLAG OFFSET FROM UN.BUF
000777                  OFFMSK  := 777                  ; MASK FOR PAGE OFFSET
000020                  OFFSET  := 16.                  ; NUMBER OF WORDS IN DOWNLINE LOAD HEADER
```

```
100000                          ONLREC  := BIT15                ; ONLINE RECOVERY IN PROGRESS FLAG
000377                          OPCOD   := LOBYT                ; MASK FOR MSCP OP CODE
001000                          PAGESZ  := 512.                 ; SIZE OF HOST PAGE
000002                          PTELEN  := 2.                   ; LENGTH OF PTE IN WORDS
140000                          Q.STAT  := 140000               ; MASK FOR MSCP PACKET QUEUE STATUS
060000                          RBNCOD  := 60000                ; RBN HEADER CODE
000077                          RBNMSK  := 77                   ; MASK FOR RBN'S/TRACK
050000                          RBNPRM  := 50000                ; PRIMARY RBN HEADER CODE
030000                          RBN2ND  := 30000                ; SECONDARY RBN HEADER CODE
110000                          H11COD  := 110000               ; BAD HEADER CODE
010000                          RBNXOR  = RBNPRM&RBN2ND         ; RBNXOR = RBNPRM .XOR. RBN2ND
060000                          RBNXOR  = RBNPRM+RBN2ND-RBNXOR-RBNXOR
010000                          H11XOR  = H11COD&RBN2ND         ; H11XOR = H11COD .XOR. RBN2ND
120000                          H11XOR  = H11COD+RBN2ND-H11XOR-H11XOR
007400                          RBNSTR  := 7400                 ; MASK FOR STARTING HI RBN
000000                          RCT.MT  := 0                    ; RCT CODE FOR EMPTY ENTRY
020000                          RCT.PR  := 20000                ; RCT CODE FOR RBN USED AS PRIMARY REPLACEMENT
030000                          RCT.SE  := 30000                ; RCT CODE FOR RBN USED AS SECONDARY REPLACEMENT
100000                          RCT.EN  := 100000               ; RCT CODE FOR ENTRY PAST END OF RCT
000017                          SDIS    := 17                   ; MASK FOR CONTROL REGISTER SDI SELECT
004400                          SDITO   := 4400                 ; SDI INTERCONNECT MAX CLOCK TICK TIMEOUT
000400                          SECSZ   := 256.                 ; SECTOR SIZE
000010                          SEQUEN  := 10                   ; SEQUENTIAL COMMAND MASK
000400                          STDALN  := BIT08                ; STANDALONE BIT IN GET STUD STATUS RESPONSE
000274                          SYNCL   := ^H0BC                ; SYNC CHARACTER LOW
023000                          SYNCH   := ^H02600              ; SYNC CHARACTER HIGH
177400                          UNITF   := HIBYT                ; MASK FOR UNIT FLAGS
000170                          CTMOUT  := 120.                 ; CONTROLLER TIMEOUT = 2 MINUTES
000440                          MODNUM  := 440                  ; CONTROLLER MODEL NUMBER (18. IN BITS 4-10)
000023                          CODVER  := 19.                  ; MICROCODE VERSION NUMBER=2.0(A000-A003=0)
                                                                ;(A004-A005=1)(A006=2)(A007=3)(A008=4)(A009=5)
                                                                ;(A010-A011=6)(A012=7)(A013-A014=8.)
                                                                ;(A015-A018=9.[mjt09])(A019=10.[mjt09a])
                                                                ;(A020=11.[mjt10],A021=11.[mjt11] A020 was
                                                                ;never released)(A022=12.[mjt12])
                                                                ;(A023=15.[mjt13])(13 & 14 not used because
                                                                ; of parity and just because it is 13.)
                                                                ;(A024=16.[E121]-[E124])
120000                          XBNCOD  := 120000               ; XBN SECTOR CODE
170000                          XBNSTR  := 170000               ; MASK FOR START XBN
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *     DEFINITION OF REFERENCES IN BUFFER MAP                                    *
                                ; *-------------------------------------------------------------------------------*
                                ; *     BASE LEVEL THREE                                                          *
                                ; *-------------------------------------------------------------------------------*
                                ; *     DIAGNOSTIC MACHINE REGISTER SAVE AREA                                     *
                                ; *-------------------------------------------------------------------------------*

000000                          DMREG7  := 0            ; DIAGNOSTIC MACHINE REGISTER 7
000001                          DMREG1  := DMREG7+1     ; DIAGNOSTIC MACHINE REGISTER 1
000002                          DMREG2  := DMREG1+1     ; DIAGNOSTIC MACHINE REGISTER 2
000003                          DMREG3  := DMREG2+1     ; DIAGNOSTIC MACHINE REGISTER 3
000004                          DMREG4  := DMREG3+1     ; DIAGNOSTIC MACHINE REGISTER 4
000005                          DMREG5  := DMREG4+1     ; DIAGNOSTIC MACHINE REGISTER 5
000006                          DMREG6  := DMREG5+1     ; DIAGNOSTIC MACHINE REGISTER 6
000007                          DMREG0  := DMREG6+1     ; DIAGNOSTIC MACHINE REGISTER 0

                                ; *-------------------------------------------------------------------------------*
                                ; *     D.PROCESSOR HEADER COMPARE STORAGE AREA                                   *
                                ; *-------------------------------------------------------------------------------*

000010                          HEADER  := DMREG0+1     ; BUFFER STORAGE FOR SECTOR HEADER (HDRCMP)

000000                                  ASSUME  HEADER&17,LE,10
                                ; *-------------------------------------------------------------------------------*
                                ; *     DRIVE RESPONSE STORAGE AREA                                               *
                                ; *-------------------------------------------------------------------------------*

000011                          INPUT   := HEADER+1     ; DRIVE RESPONSE INPUT AREA
000007                          INPLEN  := 7            ; LENGTH OF DRIVE RESPONSE AREA (14 BYTES)

                                ; *-------------------------------------------------------------------------------*
                                ; *     HOST COMMUNICATION DATA AREA                                              *
                                ; *-------------------------------------------------------------------------------*

                                ; ***   NOTE THAT RSPLEN,RSPPTR,RSPCOF,RSPPOF,RCSRW0,RCSRW1 MUST BE KEPT IN THAT ORDER ***

000020                          RSPLEN  := HEADER+8.    ; RESPONSE RING BUFFER LENGTH
000021                          RSPPTR  := RSPLEN+1     ; HOST RESPONSE RING BUFFER POINTER
                                                        ; ADDR = RING BASE + 4*RSPPTR
000022                          RSPCOF  := RSPPTR+1     ; RESPONSE CURRENT RING OFFSET SAVE
000023                          RSPPOF  := RSPCOF+1     ; RESPONSE PREVIOUS RING OFFSET SAVE
000024                          RCSRW0  := RSPPOF+1     ; RESPONSE CSR WORD 0 SAVE
000025                          RCSRW1  := RCSRW0+1     ; RESPONSE CSR WORD 1 SAVE

000000                                  ASSUME  RCSRW1,EQ,<RSPLEN+<RCSRW1-RSPLEN>>

                                ; ***   NOTE THAT CMDLEN,CMDPTR,CMDCOF,CMDPOF,CCSRW0,CCSRW1 MUST BE KEPT IN THAT ORDER ***

000026                          CMDLEN  := RCSRW1+1     ; COMMAND RING BUFFER LENGTH
000027                          CMDPTR  := CMDLEN+1     ; HOST COMMAND RING BUFFER POINTER
                                                        ; ADDR = RING BASE + 4*(RSPLEN + CMDPTR)
000030                          CMDCOF  := CMDPTR+1     ; COMMAND CURRENT RING OFFSET SAVE
```

```
000031                          CMDPOF  := CMDCOF+1      ; COMMAND PREVIOUS RING OFFSET SAVE
000032                          CCSRWO  := CMDPOF+1      ; COMMAND CSR WORD 0 SAVE
000033                          CCSRW1  := CCSRWO+1      ; COMMAND CSR WORD 1 SAVE

000000                                  ASSUME  CCSRW1,EQ,<CMDLEN+<CCSRW1-CMDLEN>>
000002                                  RSPO    := 2     ; RESPONSE INTERRUPT RING BASE NEGATIVE OFFSET
000004                                  CMDO    := 4     ; COMMAND INTERRUPT RING BASE NEGATIVE OFFSET
000006                                  VAXO    := 6     ; VAX PURGE INTERRUPT RING BASE NEGATIVE OFFSET

                                ; *-------------------------------------------------------------------------*
                                ; *     MISCELLANEOUS WORD ALLOCATIONS  (NAME OF ROUTINE USED IN)           *
                                ; *-------------------------------------------------------------------------*

000034                          FAILUR  := CCSRW1+1      ; INTERNAL FAILURE CODE
000035                          ALOLMT  := FAILUR+1      ; ALLOCATION LIMIT (U.ALOC)
000036                          SEX2SK  := ALOLMT+1      ; NUMBER OF SECTORS UNTIL SEEK (U.ALOC)
000040                          CNVTV1  := <FAILUR+17>&7760; U.CNVT VARIABLE #1 (2 WORDS)
000042                          CNVTV2  := CNVTV1+2      ; U.CNVT VARIABLE #2 (2 WORDS)
000044                          CNVTV3  := CNVTV2+2      ; U.CNVT VARIABLE #3 (1 WORD)
000045                          CNVTV4  := CNVTV3+1      ; U.CNVT VARIABLE #4 (1 WORD)
000046                          TEMP1   := CNVTV4+1      ; TEMPORARY STORAGE #1
000047                          TEMP2   := TEMP1+1       ; TEMPORARY STORAGE #2
000050                          TEMP3   := TEMP2+1       ; TEMPORARY STORAGE #3
000051                          SEKPRV  := TEMP3+1       ; SEEK ORDERING PREVIOUS SAVE
000052                          SEKSAV  := SEKPRV+1      ; SEEK ORDERING SAVE AREA

                                ; *-------------------------------------------------------------------------*
                                ; *     ECC ROUTINE BUFFER STORAGE                                          *
                                ; *     NOTE THAT THESE LOCATIONS OVERLAY THE AREA STARTING AT 'SEX2SK'     *
                                ; *-------------------------------------------------------------------------*

000035                          ECC9.1  := FAILUR+1      ; FIRST ECC 9 WORD BUFFER AREA (ECCC)
000046                          ECC9.2  := ECC9.1+9.     ; SECOND ECC 9 WORD AREA (ECCC)
000057                          ECC9.3  := ECC9.2+9.     ; THIRD ECC 9 WORD AREA (ECCC)
000035                          ELPN    := ECC9.1
000046                          ELPM    := ECC9.2
000046                          EVAL    := ELPM          ; EVAL OVERLAYS ELPM
000057                          ELPP    := ECC9.3
000070                          ELPO    := ELPP+9.
000070                          ELOC    := ELPO          ; ELOC OVERLAYS ELPO
000112                          SY      := ELPO+18.
000112                          SYO     := SY
000122                          SY8     := SY+8.
000061                          M       := ELPP+2
000062                          DISM    := ELPP+3
000063                          DISN    := ELPP+4
000064                          LM      := ELPP+5

                                ; *-------------------------------------------------------------------------*
                                ; *     REVECTORING/REPLACEMENT STORAGE AREA                                *
                                ; *-------------------------------------------------------------------------*

000123                          REVSTR  := SY8+1         ; START OF REVECTORING RAM ASSIGNMENTS
000123                          RVCSAV  := REVSTR        ; SAVES ORIGINAL REQUEST'S I/O PARAMETERS (5 WORDS)
```



```
000130                          RVCBNL  := RVCSAV+5      ; HOLDS ORIGINAL LBN IN ERROR (LO)
000131                          RVCBNH  := RVCBNL+1      ; HOLDS ORIGINAL LBN IN ERROR (HI)
000132                          RVCTA   := RVCBNH+1      ; HOLDS ORIGINAL TRACK ADDRESS AND REAL TIME OP CODE
000133                          REVEND  := RVCTA+1       ; END OF REVECTORING STORAGE AREA

                                ; *-------------------------------------------------------------------------*
                                ; *     DYNAMIC BUFFER ALLOCATION STACK AREA                                *
                                ; *     BIT 0 OF REGISTER RLL CONTAINS THE BUFFER LOCK FLAG AND ALONG WITH  *
                                ; *     BITS 1 -> 8 OF RLL FORM THE ODD POINTER INTO THE FOLLOWING STACK    *
                                ; *     THE STACK CONTAINS 1 ENTRY FOR EACH BUFFER AVAILABLE PLUS 1 AND     *
                                ; *     THE LAST ENTRY CONTAINS A ZERO TO INDICATE THAT THERE ARE NO        *
                                ; *     BUFFERS LEFT. EACH ACTIVE STACK ENTRY CONTAINS THE ADDRESS OF A     *
                                ; *     DATA BUFFER.                                                        *
                                ; *-------------------------------------------------------------------------*

000133                          BUFPTR  = REVEND                 ; START OF BUFFER POINTER STACK

000000                          ASSUME  REVEND&1,EQ,1            ; *** NOTE - BIT 0 MUST BE ONE
000133                          BUFP00  := BUFPTR                ; *** BUFFER STACK NULL ENTRY (= 0)
000134                          ECOUNT  := BUFP00+1              ; SAVE NUMBER OF ECC SYMBOLS IN ERROR
000135                          BUFP01  := ECOUNT+1              ; *** DATA BUFFER POINTER 1
000136                          HDLMT   := BUFP01+1              ; SEARCH LIMIT COUNTER (HDRCMP,D.IOPR)
000137                          BUFP02  := HDLMT+1               ; *** DATA BUFFER POINTER 2
000140                          UBURST  := BUFP02+1              ; UNIBUS TRANSFER BURST SIZE (IN TWO WORD PAIRS)
000141                          BUFP03  := UBURST+1              ; *** DATA BUFFER POINTER 3
000142                          CLIMIT  := BUFP03+1              ; COMMAND LIMIT WORK WORD
000143                          BUFP04  := CLIMIT+1              ; *** DATA BUFFER POINTER 4
000144                          OPCODE  := BUFP04+1              ; OP CODE SAVE FOR U.ALOC,U.BMTY,U.BFIL
000145                          BUFP05  := OPCODE+1              ; *** DATA BUFFER POINTER 5
000146                          MRQUE   := BUFP05+1              ; MAINTENANCE READ QUE HEAD
000147                          BUFP06  := MRQUE+1               ; *** DATA BUFFER POINTER 6
000150                          MWQUE   := BUFP06+1              ; MAINTENANCE WRITE QUE HEAD
000151                          BUFP07  := MWQUE+1               ; *** DATA BUFFER POINTER 7
000152                          QUESAV  := BUFP07+1              ; QUEUE STATUS AND LINK WORD SAVE AREA
000153                          BUFP08  := QUESAV+1              ; *** DATA BUFFER POINTER 8
000154                          LOGLEN  := BUFP08+1              ; ATTENTION/ERROR LOG PACKET LENGTH
000155                          BUFP09  := LOGLEN+1              ; *** DATA BUFFER POINTER 9
000156                          NSEEKS  := BUFP09+1              ; NUMBER OF SEEKS TO BE INITIATED
000157                          BUFP10  := NSEEKS+1              ; *** DATA BUFFER POINTER 10
000160                          RNGBSL  := BUFP10+1              ; RING BUFFER BASE LO
000161                          BUFP11  := RNGBSL+1              ; *** DATA BUFFER POINTER 11
000162                          RNGBSH  := BUFP11+1              ; RING BUFFER BASE HI (14 BITS)
000163                          BUFP12  := RNGBSH+1              ; *** DATA BUFFER POINTER 12
000164                          INTVEC  := BUFP12+1              ; HOST INTERRUPT VECTOR (IF EQ 0 THEN NEVER INTERRUPT)
000165                          BUFP13  := INTVEC+1              ; *** DATA BUFFER POINTER 13
000166                          DMTEMP  := BUFP13+1              ; DMDT TEMPORARY SAVE AREA
000167                          BUFP14  := DMTEMP+1              ; *** DATA BUFFER POINTER 14
000170                          CNTFLG  := BUFP14+1              ; MSCP CONTROLLER FLAGS

                                        ; *-------------------------------------------------------------*
                                        ; *     MSCP CONTROLLER FLAG BIT DEFINITIONS (LOWER BYTE)       *
                                        ; *-------------------------------------------------------------*

000020                                  CF.THS  := BIT04         ; ENABLE ERROR LOG MESSAGES
```

```
000100                          CF.MSC  := BIT06          ; ENABLE MISCELLANEOUS ERROR LOG MESSAGES
000200                          CF.ATN  := BIT07          ; ENABLE ATTENTION MESSAGES
000320                          CFLAGS  := CF.THS!CF.ATN!CF.MSC ; FLAGS SUPPORTED BY THE CONTROLLER

000170                  CON.ST  := CNTFLG                 ; CONTROLLER STATUS WORD (HI BYTE),
                                                          ; MSCP HOST FLAGS (LO BYTE)
                                ; *-------------------------------------------------------------------------*
                                ; *     CON.ST BIT DEFINITIONS  (UPPER BYTE)                                *
                                ; *-------------------------------------------------------------------------*
                                ;       := BIT15          ; NOT USED
                                ;       := BIT14          ; NOT USED
                                ;       := BIT13          ; NOT USED
                                ;       := BIT12          ; NOT USED
                                ;       := BIT11          ; NOT USED
                                ;       := BIT10          ; NOT USED
001000                          LFAIL   := BIT09          ; 0 = NOP, 1 = SEND LAST FAILURE LOG PKT
000400                          VAX.PG  := BIT08          ; 0 = NO PURGE, 1 = VAX PURGE REQUIRED

000171                  BUFP15  := CNTFLG+1               ; *** DATA BUFFER POINTER 15
000172                  SAVR7   := BUFP15+1               ; FOR SAVING R7 IN ECC ROUTINE
000173                  BUFP16  := SAVR7+1                ; *** DATA BUFFER POINTER 16

                        ; *-------------------------------------------------------------------------*
                        ; *     TIMER/COUNTER SERDES SYNC STORAGE                                   *
                        ; *-------------------------------------------------------------------------*

000174                  TMR.MC  := BUFP16+1               ; HOST MSCP ACTIVITY TIMER
000175                  BUFP17  := TMR.MC+1               ; *** DATA BUFFER POINTER 17
000176                  TMR.BS  := BUFP17+1               ; CONTROLLER INTERNAL TIMER BASE
000177                  BUFP18  := TMR.BS+1               ; *** DATA BUFFER POINTER 18
000200                  HOSTMO  := BUFP18+1               ; HOST TIMEOUT (SECS), IF EQ 0 THEN NO TIMEOUTS EVER !
000201                  BUFP19  := HOSTMO+1               ; *** DATA BUFFER POINTER 19
000202                  SYNC    := BUFP19+1               ; SYNC PATTERN (LEVOWR,D.WDAT)
000203                  BUFP20  := SYNC+1                 ; *** DATA BUFFER POINTER 20

                        ; *-------------------------------------------------------------------------*
                        ; *     HOST MSCP PACKET RING BUFFER STORAGE AND POINTERS                   *
                        ; *-------------------------------------------------------------------------*

000204                  HSTQUE  := BUFP20+1               ; HOST MSCP RESPONSE QUEUE
                        ;       := BITS 11 - 0            ; HOST MSCP PACKET ADDRESS
000205                  BUFP21  := HSTQUE+1               ; *** DATA BUFFER POINTER 21
000206                  DTEMP1  := BUFP21+1               ; D.PROC TEMP STORAGE
000207                  BUFP22  := DTEMP1+1               ; *** DATA BUFFER POINTER 22
000210                  UTEMP1  := BUFP22+1               ; U.PROC TEMP STORAGE
000211                  BUFP23  := UTEMP1+1               ; *** DATA BUFFER POINTER 23
000212                  LV1SV1  := BUFP23+1               ; LEV1RD STATUS SAVE WORD
000213                  BUFP24  := LV1SV1+1               ; *** DATA BUFFER POINTER 24
000214                  LV1SV2  := BUFP24+1               ; LEV1RD OPCODE SAVE WORD
000215                  BUFP25  := LV1SV2+1               ; *** DATA BUFFER POINTER 25
000216                  RVCIND  := BUFP25+1               ; HOLD ENTRY INDICATOR (FOR U.RVS7)
000217                  BUFP26  := RVCIND+1               ; *** DATA BUFFER POINTER 26
000220                  RVCRTY  := BUFP26+1               ; RETRY COUNT WHEN READING RCT
000221                  BUFP27  := RVCRTY+1               ; *** DATA BUFFER POINTER 27
```



```
000222                  RCTOFF  := BUFP27+1              ; STARTING SEARCH OFFSET IN CURRENT RCT BLOCK
000223                  BUFP28  := RCTOFF+1              ; *** DATA BUFFER POINTER 28
000224                  RCTBLK  := BUFP28+1              ; CURRENT RCT BLOCK (AS OFFSET FROM RCT BASE)
000225                  BUFP29  := RCTBLK+1              ; *** DATA BUFFER POINTER 29
000226                  RVCBUF  := BUFP29+1              ; POINTER TO ORIGINAL BUFFER IN ERROR
000227                  BUFP30  := RVCBUF+1              ; *** DATA BUFFER POINTER 30
000230                  RVCEBF  := BUFP30+1              ; HOLDS BUF.NL OF BUFFER IN ERROR
000231                  BUFP31  := RVCEBF+1              ; *** DATA BUFFER POINTER 31
000232                  RVCSDI  := BUFP31+1              ; CURRENT SDI PORT UNDERGOING REVECTORING (0 IF NONE)
000233                  BUFP32  := RVCSDI+1              ; *** DATA BUFFER POINTER 32
000234                  RVCVEC  := BUFP32+1              ; REVECTORING DISPATCH VECTOR = EVEN FOR DPROC, ODD FOR UPROC
000235                  BUFP33  := RVCVEC+1              ; *** DATA BUFFER POINTER 33
000236                  BTCNT   := BUFP33+1              ; BUFFER SERVICE FAIRNESS COUNTER
000237                  BUFP34  := BTCNT+1               ; *** DATA BUFFER POINTER 34
000240                  LN      := BUFP34+1              ; ECC VARIABLE
000241                  BUFP35  := LN+1                  ; *** DATA BUFFER POINTER 35
000242                  CMPBUF  := BUFP35+1              ;[ch04] POINTER TO DEDICATED COMPARISON BUFFER
000243                  BUFP36  := CMPBUF+1              ; *** DATA BUFFER POINTER 36
000244                  MAPSAV  := BUFP36+1              ;[UQA] TEMP SAVE FOR MAPPING INFORMATION
000245                  BUFP37  := MAPSAV+1              ; *** DATA BUFFER POINTER 37
000246                  MAPFLG  := BUFP37+1              ;[UQA]  FLAG FOR MAPPING
000247                  BUFP38  := MAPFLG+1              ; *** DATA BUFFER POINTER 38
000250                  RVECUP  := BUFP38+1              ;[UQA]  U PROC REVECTOR VECTOR
000251                  BUFP39  := RVECUP+1              ; *** DATA BUFFER POINTER 39
000252                  RVECDP  := BUFP39+1              ;[UQA] D PROC REVECTOR VECTOR
000253                  BUFP40  := RVECDP+1              ; *** DATA BUFFER POINTER 40
000254                  RVCFLG  := BUFP40+1              ;[UQA] REVECTOR PROC FLAG
000255                  BUFP41  := RVCFLG+1              ; *** DATA BUFFER POINTER 41
000256                  TEMP    := BUFP41+1              ;[V05] TEMPORARY STORAGE
000257                  BUFP42  := TEMP+1                ; *** DATA BUFFER POINTER 42
000260                  REPSTA  := BUFP42+1              ;[EERREC]HOLD REVECTOR STATUS
000261                  BUFP43  := REPSTA+1              ; *** DATA BUFFER POINTER 43
000262                  SAVUAR  := BUFP43+1              ; Storage for UAR
000263                  BUFP44  := SAVUAR+1              ; *** DATA BUFFER POINTER 44
000264                  MSCPLN  := BUFP44+1              ;[rae02] *** MSCP msg length
000265                  BUFP45  := MSCPLN+1              ; *** DATA BUFFER POINTER 45
000266                  SAVEDC  := BUFP45+1              ; [qda] Storage for edc for nonblock
000265                   BADRL  := BUFP45                ;CONTAINS LO WORD OF CONTROLLER BIIC BASE ADDRESS
000266                   BADRH  := BUFP45+1              ;CONTAINS HI WORD OF CONTROLLER BIIC BASE ADDRESS
000267                  BUFP46  := SAVEDC+1              ; *** DATA BUFFER POINTER 46
000270                  SAVBUF  := BUFP46+1              ; [qda] Storage for buffer ptr for nonblock
000271                  BUFP47  := SAVBUF+1              ; *** DATA BUFFER POINTER 47
000272                  SAVCNT  := BUFP47+1              ; [qda] Storage for word count for nonblock
000271                   RGDATL := BUFP47                ;CONTAINS LO WORD OF BIIC REGISTER DATA
000272                   RGDATH := BUFP47+1              ;CONTAINS HI WORD OF BIIC REGISTER DATA
000273                  BUFP48  := SAVCNT+1              ; *** DATA BUFFER POINTER 48
000274                  SAVADR  := BUFP48+1              ; [qda] Storage for host adr for nonblock
                         ;      := BUFP48               ;available for BDA uses
                         ;      := BUFP48+1             ;available for BDA uses
000275                  BUFP49  := SAVADR+1              ; *** DATA BUFFER POINTER 49
000276                  LGCKSV  := BUFP49+1              ; [rae07] Storage for Antilog/log check pointer
000277                  BUFP50  := LGCKSV+1              ; *** DATA BUFFER POINTER 50
000300                  BIRTRY  := BUFP50+1              ; [mjt08] bi retry count (4096)
000301                  BUFP51  := BUFP50+2              ; *** DATA BUFFER POINTER 51
```

```
000303                          BUFP52  := BUFP51+2             ; *** DATA BUFFER POINTER 52

                                ; *-------------------------------------------------------------------------*
                                ; *     MSCP PACKET STORAGE AND DEFINITION AREA (MSCP V1.2 09-APR-82)       *
                                ; *-------------------------------------------------------------------------*

000024                          NPKTS   := CMDLIM               ; NUMBER OF MSCP PACKETS
000000                          VC.CMD  := 0                    ; COMMAND VIRTUAL CIRCUIT ID
000000                          VC.RSP  := 0                    ; RESPONSE AND ATTENTION VIRTUAL CIRCUIT ID
000020                          VC.LOG  := 1*16.                ; ERROR LOG VIRTUAL CIRCUIT ID
177760                          VC.DMM  := 177760               ; DIAGNOSTIC MACHINE MODE VIRTUAL CIRCUIT ID
000002                          P       := 2                    ; MSCP PKT OFFSET (1ST WORD IS QUEUE WORD)
                                                                ; 2ND WORD IS VIRTUAL CIRCUIT IDENTIFIER
000040                          AVL.LN  := 32.                  ; AVAILABLE ATTENTION MESSAGE LENGTH (BYTES)
000034                          BAD.LN  := 28.                  ; BUS ADDRESS ERROR LOG PACKET LENGTH (BYTES)
000030                          CNT.LN  := 24.                  ; CONTROLLER ERROR LOG PACKET LENGTH (BYTES)
000054                          DSK.LN  := 44.                  ; DISK ERROR LOG PACKET LENGTH (BYTES)
000014                          DUP.LN  := 12.                  ; DUPLICATE ATTENTION MESSAGE LENGTH (BYTES)
000034                          LOG.LN  := 28.                  ; ERROR LOG PACKET MAXIMUM LENGTH (28. WORDS)
000020                          MCP.IO  := 16.                  ; LENGTH FOR MSCP I/O END PACKETS
000030                          MCP.LN  := 24.                  ; MSCP PACKET LENGTH (WORDS)
000022                          MCP.RD  := 18.                  ; MAXIMUM MSCP PACKET READ LENGTH (WORDS)
000070                          SDI.LN  := 56.                  ; SDI ERROR LOG PACKET LENGTH (BYTES)
000304                          LOGPKT  := BUFP52+1             ; COMMON ERROR LOG AND ATTENTION PACKET
000342                          LOGEND  := LOGPKT+LOG.LN+P      ; END OF LOG PACKET (27 WORDS LONG)
000342                          PKTBUF  := <LOGEND+1>&177776    ; START OF MSCP PACKET BUFFERS (MAKE SURE ITS EVEN)
000342                          PKT001  := PKTBUF               ; 1ST  MSCP PACKET BUFFER
000374                          PKT002  := PKT001+MCP.LN+P      ; 2ND  MSCP PACKET BUFFER
000426                          PKT003  := PKT002+MCP.LN+P      ; 3RD  MSCP PACKET BUFFER
000460                          PKT004  := PKT003+MCP.LN+P      ; 4TH  MSCP PACKET BUFFER
000512                          PKT005  := PKT004+MCP.LN+P      ; 5TH  MSCP PACKET BUFFER
000544                          PKT006  := PKT005+MCP.LN+P      ; 6TH  MSCP PACKET BUFFER
000576                          PKT007  := PKT006+MCP.LN+P      ; 7TH  MSCP PACKET BUFFER
000630                          PKT008  := PKT007+MCP.LN+P      ; 8TH  MSCP PACKET BUFFER
000662                          PKT009  := PKT008+MCP.LN+P      ; 9TH  MSCP PACKET BUFFER
000714                          PKT010  := PKT009+MCP.LN+P      ; 10TH MSCP PACKET BUFFER
000746                          PKT011  := PKT010+MCP.LN+P      ; 11TH MSCP PACKET BUFFER
001000                          PKT012  := PKT011+MCP.LN+P      ; 12TH MSCP PACKET BUFFER
001032                          PKT013  := PKT012+MCP.LN+P      ; 13TH MSCP PACKET BUFFER
001064                          PKT014  := PKT013+MCP.LN+P      ; 14TH MSCP PACKET BUFFER
001116                          PKT015  := PKT014+MCP.LN+P      ; 15TH MSCP PACKET BUFFER
001150                          PKT016  := PKT015+MCP.LN+P      ; 16TH MSCP PACKET BUFFER
001202                          PKT017  := PKT016+MCP.LN+P      ; 17TH MSCP PACKET BUFFER
001234                          PKT018  := PKT017+MCP.LN+P      ; 18TH MSCP PACKET BUFFER
001266                          PKT019  := PKT018+MCP.LN+P      ; 19TH MSCP PACKET BUFFER
001320                          PKT020  := PKT019+MCP.LN+P      ; 20TH MSCP PACKET BUFFER
001352                          PKTEND  := PKT020+MCP.LN+P      ; END OF MSCP PACKETS BUFFER

                                ; *-------------------------------------------------------------------------*
                                ; *     MSCP PACKET QUEUE POINTERS                                          *
                                ; *-------------------------------------------------------------------------*

                                ; THE PACKET QUEUE POINTERS PRECEED THE FIRST WORD OF THE MSCP PACKET AREA
                                ; FOR EACH OF PACKETS. THERE ARE 'NPKTS' PACKETS, 'NPKTS' ARE USED TO
```



```
                                ; HOLD MSCP COMMANDS AND 1 IS DEDICATED FOR USE WITH 'ATTENTION' AND
                                ; 'ERROR LOG' PACKETS SENT FROM THE CONTROLLER. THIS DEDICATED 'LOG' PACKET
                                ; IS ALSO USED TO HOLD AN IMMEDIATE COMMAND WHEN THE HOST HAS ALL OF THE
                                ; OTHER PACKETS IN USE. THE QUEUE POINTERS CONSIST OF A NEXT PACKET POINTER
                                ; AND STATUS BITS (P.LINK) AS DESCRIBED BELOW:

                                        ; *-----------------------------------------------------------------*
                                        ; *     RELATIVE REFERENCES TO FLAGS WITHIN A QUEUE POINTER          *
                                        ; *-----------------------------------------------------------------*

100000                                  PSTAT   := BIT15        ; 0 = MSCP PACKET AVAILABLE, 1 = MSCP PKT IN USE
040000                                  PACTV   := BIT14        ; 0 = PACKET IN QUE, 1 = PACKET ACTIVE
                                        ;       := BITS 13 - 0  ; 0=LAST PACKET IN CHAIN, NEQ 0=POINTER TO NEXT PACKET

                                        ; *-----------------------------------------------------------------*
                                        ; *     MSCP PACKET COMMAND OP CODES - MSCP VERSION 1.0 28-JAN-81    *
                                        ; *-----------------------------------------------------------------*

000001                                  OP.ABO  :=  1           ; MSCP - ABORT OP CODE
000020                                  OP.ACC  := 20           ; MSCP - ACCESS OP CODE
000017                                  OP.ATT  := 17           ; CONTROLLER - ATTENTION PROCESSING OP CODE
000010                                  OP.AVL  := 10           ; MSCP - AVAILABLE OP CODE
000021                                  OP.CCD  := 21           ; MSCP - COMPARE CONTROLLER DATA OP CODE(NOP)
000040                                  OP.CMP  := 40           ; MSCP - COMPARE HOST DATA OP CODE
000013                                  OP.DAP  := 13           ; MSCP - DETERMINE ACCESS PATHS COMMAND
000022                                  OP.ERS  := 22           ; MSCP - ERASE OP CODE
000023                                  OP.FLU  := 23           ; MSCP - FLUSH CACHE OP CODE(NOP)
000002                                  OP.GCS  :=  2           ; MSCP - GET COMMAND STATUS
000016                                  OP.GST  := 16           ; CONTROLLER - GET DRIVE STATUS OP CODE
000003                                  OP.GUS  :=  3           ; MSCP - GET UNIT STATUS OP CODE
000030                                  OP.MRD  := 30           ; CONTROLLER - MAINTENANCE READ COMMAND
000031                                  OP.MWR  := 31           ; CONTROLLER - MAINTENANCE WRITE COMMAND
000011                                  OP.ONL  := 11           ; MSCP - ONLINE OP CODE
000041                                  OP.RD   := 41           ; MSCP - READ OP CODE
000024                                  OP.RPL  := 24           ; MSCP - REPLACE OP CODE
000004                                  OP.SCC  :=  4           ; MSCP - SET CONTROLLER CHARACTERISTICS OP CODE
000012                                  OP.SUC  := 12           ; MSCP - SET UNIT CHARACTERISTICS OP CODE
000042                                  OP.WR   := 42           ; MSCP - WRITE OP CODE

                                        ; *-----------------------------------------------------------------*
                                        ; *     MSCP COMMAND PACKET OFFSETS                                 *
                                        ; *-----------------------------------------------------------------*

000000                                  P.LINK  := P-2          ; PACKET STATUS AND LINK WORD
000001                                  P.VCID  := P-1          ; VIRTUAL CIRCUIT IDENTIFIER
000000                                  UQ.CNT  := P.LINK       ; UQ PORT MSG LENGTH OVERLAID BY PACKET LINK          ;[E121]

                                                ;LOW 4 BITS OF P.VCID ARE CLEARED BY U.RECV & ARE USED AS FLAGS

000001                                          PABRT   := BIT00        ; 0 = OK, 1 = PACKET MARKED FOR ABORT
000002                                          PONER   := BIT01        ; 0 = OK, 1 = AVAIL PART OF FAILED ONLINE COMMAND
                                                ;       := BITS 3-2     ; UNUSED

000002                                  P.CRFO  := P+0          ; COMMAND REFERENCE NUMBER (LO)
```

```
000003                          P.CRF1  := P.CRFO+1     ; COMMAND REFERENCE NUMBER (HI)
000004                          P.UNIT  := P+2          ; UNIT NUMBER FIELD
000005                          P.RSO3  := P+3          ; RESERVED THIRD WORD
000004                          P.CONS  := P+2          ; DMDT CONSOLE ADDRESS (=0 FOR 777560)
000006                          P.OPCD  := P+4          ; OPCODE FIELD (LO BYTE)/HI BYTE=RESERVED
000007                          P.MOD   := P+5          ; MODIFIERS FIELD
000010                          P.RSO6  := P+6          ; RESERVED SIXTH WORD
000010                          P.BCNO  := P+6          ; BYTE COUNT (LO)
000011                          P.BCN1  := P.BCNO+1     ; BYTE COUNT (HI)
000012                          P.RSO8  := P+8.         ; RESERVED EIGHTH WORD
000012                          P.BUFO  := P+8.         ; BUFFER DESCRIPTOR (1ST WORD) LO 16 BITS OF ADDR
000013                          P.BUF1  := P.BUFO+1     ; BUFFER DESCRIPTOR (2ND WORD) HI 2 BITS OF ADDR
000014                          P.BUF2  := P.BUF1+1     ; BUFFER DESCRIPTOR (3RD WORD)
000015                          P.BUF3  := P.BUF2+1     ; BUFFER DESCRIPTOR (4TH WORD)
000016                          P.BUF4  := P.BUF3+1     ; BUFFER DESCRIPTOR (5TH WORD) SAVE LO LBN OF 1ST BAD BLOCK
000017                          P.BUF5  := P.BUF4+1     ; BUFFER DESCRIPTOR (6TH WORD) SAVE HI LBN OF 1ST BAD BLOCK
000012                          P.BUFL  := P.BUFO       ; CONTROLLER/MSCP BUFFER DESCRIPTOR LO
000013                          P.BUFH  := P.BUF1       ; CONTROLLER/MSCP BUFFER DESCRIPTOR HI
000020                          P.LBNO  := P+14.        ; LOGICAL BLOCK NUMBER (LO)
000021                          P.LBN1  := P.LBNO+1     ; LOGICAL BLOCK NUMBER (HI)
000010                          P.RFNO  := P+6          ; ABORT/GET COMMAND STATUS REF # (LO)
000011                          P.RFN1  := P.RFNO+1     ; ABORT/GET COMMAND STATUS REF # (HI)
000011                          P.UNFL  := P+7          ; ONLINE/SET UNIT CHAR UNIT FLAGS
000022                          P.SHUN  := P+16.        ; SHADOW UNIT
000022                          P.RS16  := P+16.        ; reserved field at offset 16.  ;[E122]
000023                          P.RS17  := P+17.        ; reserved field at offset 17.  ;[E122]
000012                          P.HSTO  := P+8.         ; HOST ID (LO)
000013                          P.HST1  := P.HSTO+1     ; HOST ID (HI)
000020                          P.ELGO  := P+14.        ; ERROR LOG FLAGS (LO)
000021                          P.ELG1  := P.ELGO+1     ; ERROR LOG FLAGS (HI)
000010                          P.RBNO  := P+6.         ; REPLACE RBN (LO)
000011                          P.RBN1  := P.RBNO+1     ; REPLACE RBN (HI)
000010                          P.VRSN  := P+6          ; SET CONTROLLER CHAR MSCP VERSION
000011                          P.CNTF  := P+7          ; SET CONTROLLER CHAR CONTROLLER FLAGS
000012                          P.HTMO  := P+8.         ; SET CONTROLLER CHAR HOST TIMEOUT
000020                          P.RGIO  := P+14.        ; MAINTENANCE READ/WRITE REGION ID (LO)
000021                          P.RGI1  := P.RGIO+1     ; MAINTENANCE READ/WRITE REGION ID (HI)
000022                          P.RGOO  := P+16.        ; MAINTENANCE READ/WRITE REGION OFFSET (LO)
000023                          P.RGO1  := P.RGOO+1     ; MAINTENANCE READ/WRITE REGION OFFSET (HI)
000026                          P.CYLS  := P+20.        ; GET UNIT STATUS CYLINDER SIZE

                        ; ***   THESE OFFSETS ARE TO STORE SDI I/O INFO AT THE END OF I/O PACKETS ***

000022                          S.ADRL  := P+16.        ; CURRENT HOST MEMORY TRANSFER ADDRESS (LO)
000023                          S.ADRH  := S.ADRL+1     ; CURRENT HOST MEMORY TRANSFER ADDRESS (HI)
000024                          S.CYLL  := P+18.        ; SDI LO CYLINDER #
000025                          S.CYLH  := S.CYLL+1     ; SDI HI CYLINDER #
000026                          S.GRUP  := S.CYLH+1     ; SDI GROUP NUMBER
000016                          S.LBNL  := P.BUF4       ; CURRENT HEADER (LBN) LO - OVERLAYS P.BUF4
000017                          S.LBNH  := P.BUF5       ; CURRENT HEADER (LBN) HI - OVERLAYS P.BUF5
000027                          S.TRAK  := S.GRUP+1     ; SDI TRACK NUMBER
000030                          S.SECS  := S.TRAK+1     ; SDI START SECTOR
000031                          S.SECI  := S.SECS+1     ; SDI INDEX SECTOR
000031                          S.OPFL  := S.SECI       ; OP CODE AND FLAGS FOR READ/WRITE COMPARE
```



```
                                ; *-------------------------------------------------------------------*
                                ; *     MSCP END PACKET OFFSETS                                       *
                                ; *-------------------------------------------------------------------*

000006                          P.FLGS  := P+4          ; FLAGS
000007                          P.STS   := P+5          ; END PACKET STATUS
000010                          P.MLUN  := P+6          ; MULTI-UNIT CODE
000020                          P.FBBO  := P+14.        ; FIRST BAD BLOCK (LO)
000021                          P.FBB1  := P.FBBO+1     ; FIRST BAD BLOCK (HI)
000012                          P.CMSO  := P+8.         ; COMMAND STATUS (LO)
000013                          P.CMS1  := P.CMSO+1     ; COMMAND STATUS (HI)
000014                          P.UNTO  := P+10.        ; GET UNIT STATUS UNIT IDENTIFIER (1ST WORD)
000015                          P.UNT1  := P.UNTO+1     ; GET UNIT STATUS UNIT IDENTIFIER (2ND WORD)
000016                          P.UNT2  := P.UNT1+1     ; GET UNIT STATUS UNIT IDENTIFIER (3RD WORD)
000017                          P.UNT3  := P.UNT2+1     ; GET UNIT STATUS UNIT IDENTIFIER (4TH WORD)
000023                          P.SHST  := P+17.        ; GET UNIT STATUS SHADOW STATUS
000024                          P.TRCK  := P+18.        ; GET UNIT STATUS TRACK SIZE
000025                          P.GRP   := P+19.        ; GET UNIT STATUS GROUP SIZE
000026                          P.CYL   := P+20.        ; GET UNIT STATUS CYLINDER SIZE
000027                          P.USVR  := P+21.        ; GET UNIT STATUS UNIT S/W & H/W VERSION NUMBERS
000030                          P.RCTS  := P+22.        ; GET UNIT STATUS RCT TABLE SIZE
000031                          P.RBNS  := P+23.        ; GET UNIT STATUS RBNS/TRACK
000031                          P.RCTC  := P+23.        ; GET UNIT STATUS RCT COPIES
000020                          P.MED1  := P+14.        ; 1ST WORD OF MEDIA TYPE
000021                          P.MED2  := P.MED1+1     ; 2ND WORD OF MEDIA TYPE
000024                          P.UNSO  := P+18.        ; ONLINE & SET UNIT CHAR UNIT SIZE (LO)
000025                          P.UNS1  := P.UNSO+1     ; ONLINE & SET UNIT CHAR UNIT SIZE (HI)
000026                          P.VSEO  := P+20.        ; ONLINE & SET UNIT CHAR VOL SERIAL NUMBER (LO)
000027                          P.VSE1  := P.VSEO+1     ; ONLINE & SET UNIT CHAR VOL SERIAL NUMBER (HI)
000025                          P.RS19  := P+19.        ; RESERVED NINETEENTH WORD
000012                          P.CTMO  := P+8.         ; SET CONTROLLER CHAR CONTROLLER TIMEOUT
000013                          P.CSVR  := P+9.         ; SET CONTROLLER CHAR CONTROLLER S/W & H/W VERSION #S
000014                          P.CNTO  := P+10.        ; SET CONTROLLER CHAR CONTROLLER ID WORD 1
000015                          P.CNT1  := P.CNTO+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 2
000016                          P.CNT2  := P.CNT1+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 3
000017                          P.CNT3  := P.CNT2+1     ; SET CONTROLLER CHAR CONTROLLER ID WORD 4

                                ; *-------------------------------------------------------------------*
                                ; *     END PACKET STATUS CODES                                       *
                                ; *-------------------------------------------------------------------*

000037                          ST.MSK  := 37           ; STATUS/EVENT CODE MASK
000040                          ST.SUB  := 40           ; SUB CODE MULTIPLIER
000007                          ST.SHF  := 7.           ; COMMAND OFFSET RIGHT ROTATE COUNT
000000                          ST.SUC  := 0            ; SUCCESS
000001                          ST.CMD  := 1            ; INVALID COMMAND
000002                          ST.ABO  := 2            ; COMMAND ABORTED
000003                          ST.OFL  := 3            ; UNIT OFFLINE
000004                          ST.AVL  := 4            ; UNIT AVAILABLE
000005                          ST.MFE  := 5            ; MEDIA FORMAT ERROR
000006                          ST.WPR  := 6            ; UNIT WRITE PROTECTED
000007                          ST.CMP  := 7            ; COMPARE ERROR
000010                          ST.DAT  := 10           ; DATA ERROR
```

```
000011                                  ST.HST  := 11          ; HOST BUFFER ACCESS ERROR
000012                                  ST.CNT  := 12          ; CONTROLLER ERROR
000013                                  ST.DRV  := 13          ; DRIVE ERROR

                                        ; *-------------------------------------------------------------*
                                        ; *     STATUS SUB-CODE VALUES                                  *
                                        ; *-------------------------------------------------------------*

                                        ; *     SUCCESS SUB-CODE VALUES

000000                                  SC.NML  := 0*ST.SUB    ; NORMAL SUB-CODE
000040                                  SC.SDI  := 1*ST.SUB    ; SPIN DOWN IGNORED SUB-CODE
000100                                  SC.SCN  := 2*ST.SUB    ; STILL CONNECTED SUB-CODE
000400                                  SC.AOL  := 8.*ST.SUB   ; ALREADY ONLINE SUB-CODE

                                        ; *     UNIT OFFLINE SUBCODE VALUES

000000                                  SC.UNK  := 0*ST.SUB    ; UNKNOWN UNIT SUB-CODE
000040                                  SC.NVL  := 1*ST.SUB    ; NO VOLUME MOUNTED OR DRIVE RUN/STOP SWITCH OUT
000100                                  SC.IOP  := 2*ST.SUB    ; UNIT INOPERATIVE
000400                                  SC.DIS  := 8.*ST.SUB   ; UNIT DISABLED BY FIELD SERVICE
000200                                  SC.DUP  := 4*ST.SUB    ; UNIT IS A DUPLICATE (SUCCESS SUBCODE ALSO)

                                        ; *     WRITE PROTECTED SUB-CODE VALUES

020000                                  SC.WPH  := BIT13       ; HARDWARE WRITE PROTECTED
010000                                  SC.WPS  := BIT12       ; SOFTWARE WRITE PROTECTED

                                        ; *     DATA ERROR SUB-CODE VALUES (SC.ECC USED FOR MEDIA FORMAT ALSO

000000                                  SC.FER  := 0*ST.SUB    ; FORCED ERROR
000100                                  SC.HDR  := 2*ST.SUB    ; HEADER COMPARE ERROR
000140                                  SC.DSY  := 3*ST.SUB    ; DATA SYNC TIMEOUT
000340                                  SC.ECC  := 7*ST.SUB    ; ECC ERROR (UNCORRECTABLE)
000400                                  SC.EC1  := 8.*ST.SUB   ; ONE SYMBOL ECC ERROR
000440                                  SC.EC2  := 9.*ST.SUB   ; TWO SYMBOL ECC ERROR
000500                                  SC.EC3  := 10.*ST.SUB  ; THREE SYMBOL ECC ERROR
000540                                  SC.EC4  := 11.*ST.SUB  ; FOUR SYMBOL ECC ERROR
000600                                  SC.EC5  := 12.*ST.SUB  ; FIVE SYMBOL ECC ERROR
000640                                  SC.EC6  := 13.*ST.SUB  ; SIX SYMBOL ECC ERROR
000700                                  SC.EC7  := 14.*ST.SUB  ; SEVEN SYMBOL ECC ERROR
000740                                  SC.EC8  := 15.*ST.SUB  ; EIGHT SYMBOL ECC ERROR

                                        ; *     MEDIA FORMAT ERROR SUB-CODES

000240                                  SC.N12  := 5*ST.SUB    ; NOT 512 BYTE FORMAT
000300                                  SC.BAD  := 6*ST.SUB    ; DISK NOT FORMATTED

                                        ; *     DRIVE ERROR SUB-CODE VALUES

000040                                  SC.STO  := 1*ST.SUB    ; SDI RESPONSE TIMEOUT
000100                                  SC.INV  := 2*ST.SUB    ; INVALID SDI RESPONSE
000140                                  SC.POS  := 3*ST.SUB    ; POSITIONER ERROR
000200                                  SC.RWR  := 4*ST.SUB    ; LOST READ/WRITE READY
```



```
000240                                  SC.DCL  := 5*ST.SUB    ; LOST DRIVE CLOCK OPERATION
000300                                  SC.RRD  := 6.*ST.SUB   ; LOST DRIVE RECEIVER READY
000340                                  SC.DDE  := 7.*ST.SUB   ; DRIVE DETECTED ERROR
000400                                  SC.LVO  := 8.*ST.SUB   ; RTDS PULSE ERROR/RTDS PARITY ERROR/DATA PULSE ERROR

                                        ; *     CONTROLLER ERROR SUB-CODE VALUES

000040                                  SC.OVR  := 1.*ST.SUB   ; SERDES OVERRUN ERROR
000100                                  SC.EDC  := 2.*ST.SUB   ; [ECO#1]EDC ERROR
000140                                  SC.CNT  := 3.*ST.SUB   ; [ECO#1]INCONSISTENT CONTROLLER STATE

                                        ; *     BUS ERROR SUB-CODE VALUES

000040                                  SC.ODT  := 1*ST.SUB    ; ODD TRANSFER ADDRESS
000100                                  SC.ODB  := 2*ST.SUB    ; ODD BYTE COUNT
000140                                  SC.NOM  := 3*ST.SUB    ; UNIBUS NONEXISTANT MEMORY ERROR
000200                                  SC.PAR  := 4*ST.SUB    ; UNIBUS PARITY ERROR
000240                                  sc.imr  := 5*st.sub    ; INVALID MAPPING REGISTER

                                        ; *-------------------------------------------------------------*
                                        ; *     MSCP ERROR LOG ATTENTION PACKET AND MESSAGE OFFSETS     *
                                        ; *-------------------------------------------------------------*

000002                                  L.CRFO  := P+0         ; LO COMMAND REFERENCE NUMBER
000003                                  L.CRF1  := L.CRFO+1    ; HI COMMAND REFERENCE NUMBER
000004                                  L.UNIT  := P+2         ; UNIT NUMBER
000005                                  L.SEQ   := P+3         ; SEQUENCE NUMBER
000006                                  L.FMT   := P+4         ; FORMAT
000006                                  L.FLGS  := P+4         ; FLAGS
000007                                  L.EVNT  := P+5         ; EVENT CODE
000010                                  L.CNTO  := P+6         ; CONTROLLER ID WORD 1
000011                                  L.CNT1  := L.CNTO+1    ; CONTROLLER ID WORD 2
000012                                  L.CNT2  := L.CNT1+1    ; CONTROLLER ID WORD 3
000013                                  L.CNT3  := L.CNT2+1    ; CONTROLLER ID WORD 4
000014                                  L.CSVR  := P+10.       ; CONTROLLER SOFTWARE REVISION (LO)
000014                                  L.CHVR  := L.CSVR      ; CONTROLLER HARDWARE REVISION (HI)
000015                                  L.MLUN  := P+11.       ; MULTI-UNIT CODE
000016                                  L.BADO  := P+12.       ; HOST MEMORY ADDRESS LO
000017                                  L.BAD1  := L.BADO+1    ; HOST MEMORY ADDRESS HI
000016                                  L.UNTO  := P+12.       ; UNIT ID WORD 1
000017                                  L.UNT1  := L.UNTO+1    ; UNIT ID WORD 2
000020                                  L.UNT2  := L.UNT1+1    ; UNIT ID WORD 3
000021                                  L.UNT3  := L.UNT2+1    ; UNIT ID WORD 4
000022                                  L.USVR  := P+16.       ; UNIT SOFTWARE REVISION (LO)
000022                                  L.UHVR  := L.USVR      ; UNIT HARDWARE REVISION (HI)
000023                                  L.RTRY  := P+17.       ; RETRY COUNT
000024                                  L.VSEO  := P+18.       ; VOLUME SERIAL NUMBER (LO)
000025                                  L.VSE1  := L.VSEO+1    ; VOLUME SERIAL NUMBER (HI)
000026                                  L.HDRO  := P+20.       ; HEADER (LO)
000027                                  L.HDR1  := L.HDRO+1    ; HEADER (HI)
000030                                  L.SDIO  := P+22.       ; SDI STATUS WORD 0
000031                                  L.SDI1  := L.SDIO+1    ; SDI STATUS WORD 1
000032                                  L.SDI2  := L.SDI1+1    ; SDI STATUS WORD 2
000033                                  L.SDI3  := L.SDI2+1    ; SDI STATUS WORD 3
```

```
000034                          L.SDI4  := L.SDI3+1     ; SDI STATUS WORD 4
000035                          L.SDI5  := L.SDI4+1     ; SDI STATUS WORD 5

                                ; *-------------------------------------------------------------------*
                                ; *     LOG PACKET FLAGS                                              *
                                ; *-------------------------------------------------------------------*

040000                          LF.CON  := BIT08*256.   ; OPERATION CONTINUING FLAG
000400                          LF.SNR  := BIT00*256.   ; SEQUENCE NUMBER RESET FLAG
100000                          LF.SUC  := BIT07*256.   ; OPERATION SUCCESSFUL FLAG

                                ; *-------------------------------------------------------------------*
                                ; *     MSCP PACKET CONTROLLER TO HOST OP CODES                       *
                                ; *-------------------------------------------------------------------*

                                ; ***   VIRTUAL CIRCUIT 0       ***

000100                          OP.AVA  := 100          ; AVAILABLE ATTENTION MESSAGE
000101                          OP.DUP  := 101          ; DUPLICATE UNIT NUMBER ATTENTION MESSAGE
000102                          OP.ACP  := 102          ; ACCESS PATH ATTENTION MESSAGE
000200                          OP.END  := 200          ; END PKT FLAG OR UNRECOGNIZED COMMAND END

                                ; ***   VIRTUAL CIRCUIT 1       ***

000000                          FM.CNT  := 0            ; ERROR LOG FORMAT 1 (CONTROLLER ERROR)
000001                          FM.BAD  := 1            ; ERROR LOG FORMAT 2 (UNIBUS ACCESS ERROR)
000002                          FM.DSK  := 2            ; ERROR LOG FORMAT 3 (DISK TRANSFER ERROR)
000003                          FM.SDI  := 3            ; ERROR LOG FORMAT 4 (SDI ERROR)

                                ; *-------------------------------------------------------------------*
                                ; *     MSCP PACKET MODIFIER CODES                                    *
                                ; *-------------------------------------------------------------------*

100000                          MD.EXP  := BIT15        ; EXPRESS REQUEST MODIFIER
040000                          MD.CMP  := BIT14        ; COMPARE MODIFIER
020000                          MD.CSE  := BIT13        ; CLEAR SERIOUS EXCEPTION
020000                          MD.NOV  := BIT13        ; [CONTROLLER]NO SEEK OVERLAP MODIFIER
010000                          MD.ERR  := BIT12        ; FORCE ERROR MODIFIER
004000                          MD.SCH  := BIT11        ; SUPPRESS CACHING (HIGH)       [rae02]
002000                          MD.SCL  := BIT10        ; SUPPRESS CACHING (LOW)        [rae02]
001000                          MD.SEC  := BIT09        ; SUPPRESS ERROR CORRECTION MODIFIER
000400                          MD.SER  := BIT08        ; SUPPRESS ERROR RECOVERY MODIFIER
000200                          MD.SSH  := BIT07        ; SUPPRESS SHADOWING            [rae02]
000100                          MD.WBN  := BIT06        ; WRITE BACK (NON-VOLATILE)     [rae02]
000040                          MD.WBV  := BIT05        ; WRITE BACK (VOLATILE)         [rae02]
000020                          MD.SEQ  := BIT04        ; WRITE SHAD SET ONE UNIT AT A TIME [rae02]
000020                          MD.SHD  := BIT04        ; SHADOW UNIT SUPPLIED (ERROR ON CONTROLLER)
000002                          MD.ALL  := BIT01        ; ALL CLASS DRIVERS             [rae02]
000001                          MD.SPD  := BIT00        ; SPIN DOWN MODIFIER
000001                          MD.FEU  := BIT00        ; FLUSH ENTIRE UNIT             [rae02]
000002                          MD.VOL  := BIT01        ; VOLATILE ONLY                 [rae02]
000004                          MD.SAV  := BIT02        ; SUPPRESS AVAILABLE ATTENTION MSG MODIFIER
000001                          MD.NXU  := BIT00        ; NEXT UNIT MODIFIER
000001                          MD.RIP  := BIT00        ; ALLOW SELF DESTRUCTION MODIFIER
```



```
000004                          MD.SWP  := BIT02        ; SET WRITE PROTECT
000002                          MD.IMF  := BIT01        ; IGNORE MEDIA FORMAT ERROR MODIFIER
000001                          MD.PRI  := BIT00        ; PRIMARY REPLACEMENT MODIFIER
000001                          MD.FKC  := BIT00        ; Flush Encryption/Decryption Key Cache [ch10]
000040                          MD.EXC  := BIT05        ; Exclusive Access (unit) [ch10]

                                ; *-------------------------------------------------------------------*
                                ; *     END PACKET FLAGS                                              *
                                ; *-------------------------------------------------------------------*

100000                          EF.BBR  := BIT15        ; BAD BLOCK REPORTED
040000                          EF.BBU  := BIT14        ; BAD BLOCK UNREPORTED
020000                          EF.LOG  := BIT13        ; ERROR LOG GENERATED

                        ; *-----------------------------------------------------------------------*
                        ; *     REFERENCES WITHIN THE SDI INTERCONNECT CONTROL BLOCK               *
                        ; *-----------------------------------------------------------------------*

000000                  SDI.ST  := 0            ; STATUS OF SDI INTERCONNECT

                                ; *-------------------------------------------------------------------*
                                ; *     SDI INTERCONNECT STATUS FLAG EQUATES                          *
                                ; *-------------------------------------------------------------------*

000001                          PKIP    := BIT00        ; 0 = NO PKT, 1 = PACKET IN PROGRESS
000002                          SEEK    := BIT01        ; 0 = NO SEEK NEEDED, 1 = SEEK INITIATE NEEDED
000004                          DERR    := BIT02        ; 0 = NO ERROR, 1 = FATAL ERROR ON I/O COMMAND
000010                          VECT    := BIT03        ; 0 = VECTOR LEVEL DONE, 1 = STATE VECTOR LEVEL ACTIVE
000020                          BFRQ    := BIT04        ; 0 = OK, 1 = BUFFER CONTROL BLOCKS NEEDED
000040                          SUSP    := BIT05        ; 0 = TASK RUNNING, 1 = TRANSFER TASK SUSPENDED
000100                          BFSV    := BIT06        ; 0 = OK, 1 = BUFFER SERVICE REQUIRED FROM U.PROC
000200                          XCMP    := BIT07        ; 0 = OK, 1 = TRANSFER TASK COMPLETED
000400                          DATT    := BIT08        ; 0 = OK, 1 = DRIVE ATTENTION PENDING
001000                          GSEL    := BIT09        ; MODIFIES "SEEK"; 0=SEEK, 1=GROUP SELECT ONLY          ;[E121]
002000                          DRDUP   := BIT10        ; DRDUP, DRVOL, AND DRAVL ARE ENCODED AS FOLLOWS:
004000                          DRVOL   := BIT11        ; 000 = DRIVE ONLINE
010000                          DRAVL   := BIT12        ; 001 = DRIVE ONLINE AND DUPLICATE
                                                        ; 010 = DRIVE OFFLINE AND NOT USABLE
                                                        ; 011 = DRIVE OFFLINE AND DUPLICATE
                                                        ; 100 = DRIVE AVAILABLE (SPINABLE OR NOT SPINABLE)
                                                        ; 101 = NOT USED
                                                        ; 110 = NOT USED
                                                        ; 111 = DRIVE OFFLINE AND UNKNOWN TO CONTROLLER
020000                          SLAT    := BIT13        ; 0 = NOP, 1 = SEND LOG OR ATTENTION PKT TO HOST
                                                        ; IN ADDITION SLAT IS USED AS A FLAG TO AID IN
                                                        ; DETERMINING IF A DRIVE IS ALREADY ONLINE TO THE CONTROLLER
040000                          ERRIP   := BIT14        ; 0 = NO ERROR RECOVERY, 1 = LEVEL 0 RECOVERY ACTIVE
                                                        ; IN ADDITION ERRIP IS USED AFTER INITIALIZATION
                                                        ; TO FORCE INIT TO ALL DRIVES
100000                          RVCT    := BIT15        ; 0 = NO REVECTORING REQUIRED, 1 = REVECTORING REQUIRED
                                                        ; IF THIS BIT = 1 AND "RVCSDI" POINTS TO THIS SDI BLOCK,
                                                        ; THEN REVECTORING IS IN PROGRESS FOR THIS DRIVE.

000001                  SDI.SL  := SDI.ST+1     ; HEADER COMPARE SEARCH LIMIT
```

```
000002                          SDI.CP  := SDI.SL+1     ; POINTER TO CURRENTLY ACTIVE UNIT CHARACTERISTICS
000003                          SDI.SW  := SDI.CP+1     ; STATE WORD - MSCP END STATUS/ERROR LOG CODE
000004                          SDI.TM  := SDI.SW+1     ; SDI INTERCONNECT ACKNOWLEDGE/COMMAND TIMER
000005                          SDI.UG  := SDI.TM+1     ; U.PROC GROUP WORD (USED BY U.CPRM)
000006                          SDI.UB  := SDI.UG+1     ; ADDRESS OF U.PROC'S BUFFER CONTROL BLOCK(FOR I/O)
000007                          SDI.DB  := SDI.UB+1     ; ADDRESS OF D.PROC'S BUFFER CONTROL BLOCK(FOR I/O)

                                ; ***   NOTE - THE FOLLOWING 4 WORDS (SDI.1T,SDI.CL,SDI.CH,SDI.GP) ARE USED TO SEND
                                ; ***   THE SDI LEVEL 2 SEEK COMMAND DIRECTLY FROM THE SDI CONTROL BLOCK

000010                          SDI.1T  := SDI.DB+1     ; TEMP WORD (USED FOR ERROR LOG/ATTN CODE,SEEK COMMAND)
000011                          SDI.CL  := SDI.1T+1     ; CURRENT CYLINDER ADDRESS LO (FORMS SEEK COMMAND)
000012                          SDI.CH  := SDI.CL+1     ; CURRENT CYLINDER ADDRESS HI (FORMS SEEK COMMAND)
000013                          SDI.GP  := SDI.CH+1     ; CURRENT GROUP NUMBER  (FORMS SEEK COMMAND)
000014                          SDI.SS  := SDI.GP+1     ; # OF SECTORS TO TRANSFER BEFORE SEEK REQUIRED
000015                          SDI.XL  := SDI.SS+1     ; LO NUMBER OF BYTES REMAINING TO BE TRANSFERRED
000016                          SDI.XH  := SDI.XL+1     ; HI NUMBER OF BYTES REMAINING TO BE TRANSFERRED

                                ; ***   SDI CONTROL BLOCK ERROR STATE CONTROL WORDS ***

000017                          SDI.E1  := SDI.XH+1     ; LEVEL 1 ERROR STATE CONTROL WORD
                                                        ; USED FOR READ RETRIES (EDC AND ECC ERRORS)
000020                          SDI.E0  := SDI.E1+1     ; LEVEL 0 ERROR STATE CONTROL WORD
                                                        ; USED FOR SDI LEVEL 2,1 ERRORS
                                                        ; AND FOR READ,WRITE,COMPARE DRIVE ERROR RETRIES
000377                                  ERRVEC  := LOBYT        ; ERROR STATE VECTOR (SDI.E0 AND SDI.E1)
                                                                ; 0 = LEVEL INACTIVE, NEQ 0 = LEVEL ACTIVE
001400                                  LV2CNT  := 1400         ; SDI LEVEL TWO RETRY COUNT (2 BITS LONG)
                                                                ; INCREMENT RETRY COUNT WITH ADDC #377,REG
                                                                ; USED WITH SDI.E0
007400                                  RETCNT  := 7400         ; I/O RETRY COUNT (USED WITH SDI.E0)
036000                                  IORTY   := 36000        ; I/O RETRY INDICATOR FLAGS
002000                                          IOCNT   := BIT10        ; READ/WRITE RETRY INDICATOR (1=LAST RETRY)
004000                                          IOSEK   := BIT11        ; SEEK ERROR (RECAL,RESEEK IN RECOVERY)
010000                                          IOCLK   := BIT12        ; DRIVE CLOCK TIMEOUT ERROR (INIT,RESEEK IN RECOVERY)
020000                                          IORWR   := BIT13        ; R/W RDY OR SERDES LATE ERROR (RETRY I/O IN RECOVERY)
040000                                  ERRINI  := BIT14        ; 0 = NO INIT DONE, 1 = INIT DONE ON RECOVERY
                                                                ; USED WITH SDI.E0
100000                                  ERRINP  := BIT15        ; 0 = NO RECOVERY, 1 = RECOVERY IN PROGRESS
                                                                ; USED WITH SDI.E0 AND SDI.E1
000021                          SDI.SV  := SDI.E0+1     ; SDI LEVEL 2 FUNCTION STATE VECTOR
000022                          SDI.PQ  := SDI.SV+1     ; POINTER TO HEAD OF CONTROL BLOCK MSCP PACKET QUEUE
000023                          SDI.RO  := SDI.PQ+1     ; ROTATIONAL OPTIMIZATION COUNT
000024                          SDI.OE  := SDI.RO+1     ; SDI OVERLAP ENABLE FLAG;
                                                        ; 0 = DISABLE, 1 = ALLOW, -1 = D.PROC SAW LAST BUFFER
000025                          SDI.OM  := SDI.OE+1     ; STORAGE FOR CURRENT MSCP PKT PTR WHILE OVERLAPPING
000026                          SDI.ES  := SDI.OM+1     ; SDI EXTENDED STATUS
000001                                  RVWRIT  := BIT00        ; REVECTOR WRITE FLAG FOR U.BFLC
000002                                  OFFTRK  := BIT01        ; SEEK REQUIRED DUE TO ERROR RECOVERY LEVEL USED
000004                                  FSEEK   := BIT02        ; FORCE SEEK FLAG
                                        ;       := BIT03        ; NOT USED
000020                                  RVACTV  := BIT04        ; REVECTORING ACTIVE
000040                                  URETRY  := BIT05        ; U.PROC RETRY IN PROCESS
                                        ;       := BIT06-BIT07  ; UNUSED
```



```
                                        ;       := BIT08-BIT14  ; SEEK FAIRNESS COUNTER
077400                                  FAIRCT  := 77400        ; SEEK FAIRNESS COUNT MASK
100000                                  ELEV    := BIT15        ; SEEK ELEVATOR FLAG

000027                          SDI.BL  := SDI.ES+1     ; LO BAD BLOCK DETECTED (CLEAR IN U.ALOC)
000030                          SDI.BH  := SDI.BL+1     ; HI BAD BLOCK DETECTED (CLEAR IN U.ALOC)
000031                          SDI.RL  := SDI.BH+1     ; LO LBN OF PRIMARY RBN
000032                          SDI.RH  := SDI.RL+1     ; HI LBN OF PRIMARY RBN

                                ; ***   START OF DRIVE GENERAL CHARACTERISTICS  (SDI V3.7, 15-JUNE-81)

000033                          SDI.TO  := SDI.RH+1     ; LO BYTE = SHORT T.O. (LO)/SDI VERS (HI)
000033                          SDI.XR  := SDI.TO       ; HI BYTE = TRANSFER RATE
000034                          SDI.LT  := SDI.TO+1     ; LO BYTE = LONG T.O. (LO)/RETRY # (HI)
000034                          SDI.RC  := SDI.LT       ; HI BYTE = RCT COPIES (LO)/RESERVED (HI)
100000                                  UNC.SS  := BIT15        ; 0 = 512 BYTE ONLY, 1 = 512 OR 576 BYTE SECT
000035                          SDI.ER  := SDI.LT+1     ; LO BYTE = ERROR RECOVERY LEVELS
000035                          SDI.EC  := SDI.ER       ; HI BYTE = ECC ERROR THRESHOLD
000036                          SDI.RS  := SDI.ER+1     ; HARDWARE REVISION #(HI)/SOFTWARE REVISION #(LO)
100000                                  FDIAG   := BIT15        ; 1= LIMITED DIAG, 0 = FULL DIAG SUPPORT        ;[E121]
000037                          SDI.I1  := SDI.RS+1     ; UNIQUE DRIVE ID #1
000040                          SDI.I2  := SDI.I1+1     ; UNIQUE DRIVE ID #2
000041                          SDI.I3  := SDI.I2+1     ; UNIQUE DRIVE ID #3
000042                          SDI.TI  := SDI.I3+1     ; DRIVE TYPE IDENTIFIER (LO)/DRIVE REVS/SEC (HI)
000043                          SDI.UE  := SDI.TI+1     ; END OF CONTROLLER CRITICAL DRIVE COMMON CHARACTERISTICS
000010                          SDI.DL  := <SDI.TI+1-SDI.TO>; LENGTH OF DRIVE COMMON CHARACTERISTICS

                                ; ***   START OF SDI STATUS STORAGE (7 WORDS) ***

000043                          SDI.UN  := SDI.UE       ; SDI STATUS WORD 0 (UNIT NUMBER INFORMATION)

                                        ; *-------------------------------------------------------------------*
                                        ; *     DRIVE STATUS UNIT WORD BIT DEFINITIONS                        *
                                        ; *-------------------------------------------------------------------*

010000                                  DRV.U1  := BIT12        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
020000                                  DRV.U2  := BIT13        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
040000                                  DRV.U3  := BIT14        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
100000                                  DRV.U4  := BIT15        ; 0 = NO SUBUNIT, 1 = SUBUNIT PRESENT
170000                                  DRV.SU  := <DRV.U1+DRV.U2+DRV.U3+DRV.U4>; SUBUNIT MASK

000044                          SDI.S1  := SDI.UN+1     ; SDI STATUS WORD 1

                                        ; *-------------------------------------------------------------------*
                                        ; *     DRIVE STATUS 1ST WORD BIT DEFINITIONS                         *
                                        ; *-------------------------------------------------------------------*

000001                                  DRV.RU  := BIT00        ; 0 = RUN/STOP SWIT OUT, 1 = RUN/STOP SWIT IN
000002                                  DRV.PS  := BIT01        ; 0 = PORT SWITCH OUT, 1 = PORT SWITCH IN
                                        ;       := BIT02        ; NOT USED
000010                                  DRV.EL  := BIT03        ; 0 = NO ACTION, 1 = PLEASE SEND A LOG PACKET
000020                                  DRV.SR  := BIT04        ; 0 = SPINDLE NOT AT SPEED, 1 = SPINDLE AT SPEED
000040                                  DRV.DR  := BIT05        ; 0 = NO DIAG REQ'D, 1 = DIAGNOSTIC LOAD REQ'D
000100                                  DRV.RR  := BIT06        ; 0 = NO READJ REQ'D, 1 = READJUSTMENT REQ'D
```

```
000200                          DRV.OA  := BIT07        ; 0 = ONLINE/AVAILABLE, 1 = ONLINE TO OTHER PORT
000400                          DRV.S7  := BIT08        ; 0 = 512 BYTE SECTOR, 1 = 576 BYTE SECTOR
001000                          DRV.DB  := BIT09        ; 0 = NO DIAG CYL ACCESS, 1 = YES DIAG CYL ACCESS
002000                          DRV.FO  := BIT10        ; 0 = NO FORMATTING, 1 = FORMATTING ENABLED
004000                          DRV.DD  := BIT11        ; 0 = DRIVE ENABLED, 1 = DRIVE DISABLED FOR DIAGNOSTICS
010000                          DRV.W1  := BIT12        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
020000                          DRV.W2  := BIT13        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
040000                          DRV.W3  := BIT14        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
100000                          DRV.W4  := BIT15        ; 0 = W.P. SWITCH OUT, 1 = W.P. SWITCH IN
000045                          SDI.S2  := SDI.S1+1     ; SDI STATUS WORD 2

                                ; *-------------------------------------------------------------------*
                                ; * LOWER BYTE OF SDI.S2 IS STATUS ERROR BYTE                         *
                                ; *-------------------------------------------------------------------*
                                ;       := BIT00        ; NOT USED
                                ;       := BIT01        ; NOT USED
                                ;       := BIT02        ; NOT USED
000010                          DRV.WE  := BIT03        ; 0 = OK, 1 = WRITE LOCK ERROR
000020                          DRV.DF  := BIT04        ; 0 = INIT DIAG OK, 1 = INIT DIAG FAILURE
000040                          DRV.PE  := BIT05        ; 0 = OK, 1 = PROTOCOL ERROR
000100                          DRV.RE  := BIT06        ; 0 = OK, 1 = RETRYABLE ERROR-REISSUE COMMAND
000200                          DRV.DE  := BIT07        ; 0 = OK, 1 = DRIVE ERROR
000170                          DCMASK  := <DRV.WE+DRV.DF+DRV.PE+DRV.RE>
                                                        ; DRIVE CLEAR COMMAND ERROR CLEAR MASK
000400                          DRV.C4  := BIT08        ; CONTROLLER STATUS - IGNORED BY DRIVE
001000                          DRV.C3  := BIT09        ; CONTROLLER STATUS - IGNORED BY DRIVE
002000                          DRV.C2  := BIT10        ; CONTROLLER STATUS - IGNORED BY DRIVE
004000                          DRV.C1  := BIT11        ; CONTROLLER STATUS - IGNORED BY DRIVE
010000                          DRV.S1  := BIT12        ; CONTROLLER STATUS - IGNORED BY DRIVE
020000                          DRV.S2  := BIT13        ; CONTROLLER STATUS - IGNORED BY DRIVE
040000                          DRV.S3  := BIT14        ; CONTROLLER STATUS - IGNORED BY DRIVE
100000                          DRV.S4  := BIT15        ; CONTROLLER STATUS - IGNORED BY DRIVE
170000                          DRV.SN  := DRV.S1+DRV.S2+DRV.S3+DRV.S4 ; MASK FOR TESTING S BITS

                        ; ***   SDI CONTROL BLOCK PRIVATE ATTENTION PACKET ***
                        ; ***   NOTE - BIT 0 OF THE PACKET CONTROL WORD ADDRESS DIFFERENTIATES BETWEEN
                        ; ***   ATTENTION PACKETS AND NORMAL MSCP PACKETS, I.E. BIT 0 = 1 MEANS IT IS
                        ; ***   AN INTERNAL ATTENTION PACKET AND BIT 0 = 0 MEANS IT IS A MSCP PACKET.

000046                  SDI.AT  := SDI.S2+1             ; ATTENTION PACKET CONTROL WORD
000047                  SDI.2T  := SDI.AT+1             ; 2ND TEMP WORD - ATTENTION VIRTUAL CIRCUIT NOT NEEDED
000050                  SDI.S4  := SDI.AT+P.CRF0        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000051                  SDI.S5  := SDI.AT+P.CRF1        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000052                  SDI.S6  := SDI.AT+P.UNIT        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000053                  SDI.S7  := SDI.AT+P.RSO3        ; UNUSED IN ATTN PKT - SDI EXTENDED STATUS
000054                  SDI.OP  := SDI.AT+P.OPCD        ; USED FOR ATTENTION/GET STATUS INTERNAL OP CODE
000055                  SDI.PL  := SDI.AT+P.MOD         ; WRITE PRELOAD COUNTER (NOTE MSB MUST BE ZERO)
                        ;SDIB.L := SDI.AT+8.            ; LENGTH OF SDI CONTROL BLOCK

                        ; ***   START OF SUBUNIT CHARACTERISTICS BUFFER ***

                        ;SDI.CW := 0             ; SUBUNIT BUFFER CONTROL WORD
000056                  SDI.CW  := SDI.PL+1      ; [V05]SUBUNIT BUFFER CONTROL WORD
                                ;       BITS 15-12      ; SUBUNIT CODE (0001,0010,0100,1000)
```



```
004000                          DRV.AV  := BIT11        ; 0 = SUBUNIT ONLINE, 1 = SUBUNIT AVAILABLE
002000                          DRV.AT  := BIT10        ; 0 = ATTENTION SENT, 1 = NEED TO SEND ATTENTION
                                ;       := BITS 9-0     ; UNIT NUMBER (0 - 1023) FOR THIS SUBUNIT
176000                          DRV.UM  := DRV.SU!DRV.AT!DRV.AV; UNIT NUMBER MASK (USE WITH 'BIC')
                                ; IF SDI.CW IS ZERO THEN IT IS AVAILABLE, OTHERWISE IT IS IN USE.

000057                  SDI.SD  := SDI.CW+1     ; PARENT SDI CONTROL BLOCK POINTER
000060                  SDI.UF  := SDI.SD+1     ; UNIT FLAGS FOR MSCP PACKET

                                ; *-------------------------------------------------------------------*
                                ; *     UNIT FLAG BIT DEFINITIONS                                     *
                                ; *-------------------------------------------------------------------*

100000                          UF.RPL  := BIT15        ; 1 = NO HOST BBR ON THIS UNIT                  ;[E121]
020000                          UF.WPH  := BIT13        ; 1 = WRITE PROTECTED HARDWARE
010000                          UF.WPS  := BIT12        ; 1 = WRITE PROTECTED SOFTWARE
000200                          UF.RMV  := BIT07        ; 1 = REMOVABLE MEDIA
000004                          UF.576  := BIT02        ; 1 = VOLUME MOUNTED HAS 576 BYTE SECTORS
000002                          UF.CMW  := BIT01        ; 1 = COMPARE ON ALL WRITE OPERATIONS
000001                          UF.CMR  := BIT00        ; 1 = COMPARE ON ALL READ OPERATIONS
000003                          UF.MSK  := <UF.CMW+UF.CMR>

000061                  SDI.V1  := SDI.UF+1     ; VOLUME SERIAL NUMBER WORD 1
000062                  SDI.V2  := SDI.V1+1     ; VOLUME SERIAL NUMBER WORD 2
000063                  SDI.H1  := SDI.V2+1     ; LO ORDER LBN SPACE IN CYLINDERS
000064                  SDI.H2  := SDI.H1+1     ; HI ORDER LBN SPACE IN CYLS (LO)/HI CYL # (HI)
000065                  SDI.GC  := SDI.H2+1     ; GROUPS PER CYL (LO)//HI START LBN (LO)/HI START XBN (HI)
000066                  SDI.TG  := SDI.GC+1     ; TRACKS PER GROUP (LO)//HI START RBN (LO)/HI START DBN (HI)
000067                  SDI.RT  := SDI.TG+1     ; # REPLACEMENTS PER TRACK,RM FLAG (LO)/RESERVED (HI)

                                ; *-------------------------------------------------------------------*
                                ; *     SUBUNIT CAPABILITIES BIT DEFINITIONS                          *
                                ; *-------------------------------------------------------------------*

                                UNC.RM  := BIT07        ; 0 = NO REMOVABLE MEDIA, 1 = REMOVABLE MEDIA
000200
000070                  SDI.DP  := SDI.RT+1     ; DATA PREAMBLE SIZE (LO)/HEADER PREAMBLE SIZE (HI)
000071                  SDI.M1  := SDI.DP+1     ; MEDIA TYPE LO ORDER
000072                  SDI.M2  := SDI.M1+1     ; MEDIA TYPE HI ORDER
000073                  SDI.FC  := SDI.M2+1     ; FCT COPY SIZE IN XBNS
000074                  SDI.12  := SDI.FC+1     ; # 512 BYTE LBNS PER TRACK (LO)
000074                  SDI.GO  := SDI.12       ; GROUP OFFSET (HI)
000075                  SDI.L1  := SDI.GO+1     ; LBN'S IN HOST AREA (LO)
000076                  SDI.L2  := SDI.L1+1     ; LBN'S IN HOST AREA (HI)
000077                  SDI.RV  := SDI.L2+1     ; RCT SIZE IN LBN'S
000100                  SDI.PH  := SDI.RV+1     ;[V05] TEMP FOR PRIMARY HDR DURING REVECT
000101                  SDI.LL  := SDI.PH+1     ; LENGTH OF UNIT CHARACTERISTICS

                        ; *-------------------------------------------------------------------*
                        ; *     HOST MAPPING INFORMATION                                      *
                        ; *-------------------------------------------------------------------*

000101                  MAP.CH  := SDI.PH+1     ;PTR TO START OF PTE CACHE
000102                  MAP.NX  := MAP.CH+1     ;PTR TO CURRENT PTE ENTRY
000103                  MAP.MO  := MAP.NX+1     ;ADDRESS OF NEXT UNREAD PTE IN HOST MEM LOW
```

```
000104                          MAP.M1   := MAP.M0+1      ;ADDRESS OF NEXT UNREAD PTR IN HOST MEM HIGH
000105                          MAP.RD   := MAP.M1+1      ;NUMBER OF CACHED BUT UNUSED PTE'S
000106                          MAP.UR   := MAP.RD+1      ;NUMBER OF REMAINING UNREAD PTE'S FOR THIS REQ
000107                          MAP.OF   := MAP.UR+1      ;OFFSET INTO PAGE
000110                          MAP.S1   := MAP.OF+1      ;SEGMENT 1 SIZE
000111                          MAP.V0   := MAP.S1+1      ;LOW ORDER MAPPING REGISTER NUMBER
000112                          MAP.V1   := MAP.V0+1      ;HIGH ORDER MAPPING REGISTER NUMBER
000113                          MAP.V2   := MAP.V1+1      ;LOW ORDER MAPPING BASE ADDRESS
000114                          MAP.V3   := MAP.V2+1      ;HIGH ORDER MAPPING BASE ADDRESS
000115                          MAP.ST   := MAP.V3+1      ;STATUS -- 1 = MAP INIT DONE, 0 = NOT DONE [kjk]
000116                          MAP.ND   := MAP.ST+1      ;END OF MAPPING INFORMATION
000120                          SDIB.L   := MAP.ND+2.     ;[V05] LENGTH OF SDI CONTROL BLOCK PLUS EXTRA

                                ; *-------------------------------------------------------------------------*
                                ; *      DATA BUFFER CONTROL BLOCK DEFINITIONS                              *
                                ; *-------------------------------------------------------------------------*

000000                          BUF.NL   := 0             ; POINTER TO NEXT DATA BUFFER CONTROL BLOCK

                                        ; *-------------------------------------------------------------------*
                                        ; *      DATA BUFFER USE FLAGS (UPPER 2 BITS OF BUF.NL)               *
                                        ; *-------------------------------------------------------------------*

100000                                  BLAST    := BIT15         ; 0 = CONTINUE, 1 = LAST BUF/SEEK REQ
040000                                  BFULL    := BIT14         ; 0 = BUFFER EMPTY, 1 = BUFFER FULL

000001                          BUF.ST   := BUF.NL+1      ; BUFFER CONTROL BLOCK STATUS WORD

                                        ; *-------------------------------------------------------------------*
                                        ; *      BUFFER CONTROL BLOCK STATUS BIT DEFINITIONS                  *
                                        ; *-------------------------------------------------------------------*

100000                                  BRTRY    := BIT15         ; 0 = NORMAL, 1 = U.PROC READ RETRY
040000                                  BLSTB    := BIT14         ; 0 = NOT LAST, 1 = RETRY REALLY IS LAST BUFFER
020000                                  BECER    := BIT13         ; 0 = NORMAL, 1 = FATAL ECC ERROR, BUT GIVE BEST GUESS DATA
010000                                  BECC     := BIT12         ; 0 = NO ECC ERROR, 1 = ECC ERROR
004000                                  BGOOD    := BIT11         ; 0 = NO HEADERS OK, 1 = AT LEAST 1 HEADER OK
002000                                  BDSNF    := BIT10         ; 0 = NO DATA SYNC ERR, 1 = DATA SYNC NOT FOUND [EERREC]
001000                                  BLRWR    := BIT09         ; 0 = NO RD/WR RDY ERR, 1 = LOST RD/WR RDY [EERREC]
000400                                  BERDN    := BIT08         ; 0 = NO ERR REC CMD ISSUED, 1 = ERR REC CMD ISSSUED [EERREC]
                                        ;        := BITS 7-6      ; NOT USED
000040                                  BRCTS    := BIT05         ; 0 = RCT HAS NOT BEEN SEARCHED, 1 = HAS BEEN SEARCHED [EERREC]
000020                                  BRARS    := BIT04         ; 0 = RCT HAS NOT ALREADY BEEN SEARCHED, 1 = HAS ALREADY BEEN SEARCHED [E
000010                                  BMAPDN   := BIT03         ; 0 = MAPPING NOT DONE, 1 = MAPPING DONE
000004                                  BNXCOP   := BIT02         ; 0 = FATAL ERROR, 1 = READ NEXT RCT COPY
000002                                  BCGRP    := BIT01         ; 0 = SELECT GROUP NOT DONE, 1 = SELECT GROUP DONE
000001                                  BGRUP    := BIT00         ; 0 = NO ACTION, 1 = SELECT GROUP

000002                          BUF.BP   := BUF.ST+1      ; DATA BUFFER POINTER
000003                          BUF.HL   := BUF.BP+1      ; 1ST WORD OF EXPECTED HEADER
000004                          BUF.HH   := BUF.HL+1      ; 2ND WORD OF EXPECTED HEADER
000005                          BUF.TA   := BUF.HH+1      ; TRACK ADDRESS AND I/O COMMAND THIS SECTOR
000006                          BUF.SD   := BUF.TA+1      ; POINTER TO PARENT SDI CONTROL BLOCK
000007                          BUF.UA   := BUF.SD+1      ; HOST ADDRESS BITS 15 - 0 (FOR THIS SECTOR)
```



```
000010                          BUF.US   := BUF.UA+1      ;[BDA] BI ADDR BITS 16/30 -> BITS <0,13>
000011                          BUF.GP   := BUF.US+1      ;[BDA] GROUP FOR THIS SECTOR -> BITS <0,7>
                                                          ;[BDA] 2ND SEGMENT WORD COUNT -> BITS <8,15>
000012                          BUF.BC   := BUF.GP+1      ;[BDA] BYTE COUNT FOR U.PROC (TO/FROM HOST)
000013                          BUF.U1   := BUF.BC+1      ;2ND SEGMENT HOST ADDRESS LOW
000014                          BUF.U2   := BUF.U1+1      ;[BDA]2ND SEGMENT HOST ADDRESS HIGH -> BITS <0,13>
                                ;BUF.U3  := BUF.U2+1      ;[BDA] BYTE COUNT FOR SECOND SEGMENT -> BITS <0,8>
000015                          BUF.LL   := BUF.U2+1      ;[BDA]BUFFER CONTROL BLOCK LENGTH

                                ; *-------------------------------------------------------------------------*
                                ; *      DATA BUFFER DEFINITIONS                                            *
                                ; *-------------------------------------------------------------------------*

000000                          BUF.56   := 0             ; START OF 256 WORD DATA BUFFER
000400                          BUF.ED   := BUF.56+SECSZ  ; DATA ERROR CORRECTION CODE (EDC)
000401                          BUF.EC   := BUF.ED+1      ; START OF 12 WORDS HOLDING 17 PACKED ECC SYMBOLS
000415                          BUF.DL   := BUF.EC+12.    ; DATA BUFFER LENGTH

                                ; ********** END BUFFER MAP DEFINITIONS ************

                                ; *-------------------------------------------------------------------------*
                                ; *      SUBUNIT CHARACTERISTICS,                                           *
                                ; *      BUFFER CONTROL BLOCK,                                              *
                                ; *      SDI INTERCONNECT CONTROL BLOCK, AND                                *
                                ; *      BUFFER ALLOCATION                                                  *
                                ; *-------------------------------------------------------------------------*

000200                          NBCB     := 32.*4         ; [BDA]NUMBER OF BUFFER CONTROL BLOCKS
000051                          NBUFR    := 41.           ; [cho4]Number of data buffers that are dynamically
                                                          ;       allocated.  Buffer NBUFR+1 (ie, buffer 42)
                                                          ;       is reserved for use by the comparison routine.
000010                          NSCB     := 8.            ; NUMBER OF SUBUNIT CHARACTERISTICS BUFFERS
000004                          NSDI     := 4.            ; NUMBER OF SDI INTERCONNECTS
001352                          DM.BEG   := 1352          ; DM MACHINE MUST START AT 1352 !!!!!!!

000000                          ASSUME   PKTEND,LE,DM.BEG; MAKE SURE PACKETS DON'T INTRUDE ON DM SPACE !!!!
001355                          UN.BUF   := DM.BEG+3      ; START OF SUBUNIT CHARACTERISTICS BUFFERS
001355                          SDIBEG   = <UN.BUF&177776>+1; [V05]START OF SDI CONTROL BLOCKS (ODD)

000000                          ASSUME   <SDIBEG&BIT00>,EQ,1; MAKE SURE START IS ODD
001355                          SDI.1    := SDIBEG        ; START OF SDI INTERCONNECT #1 CONTROL BLOCK
001475                          SDI.2    := SDI.1+SDIB.L  ; START OF SDI INTERCONNECT #2 CONTROL BLOCK
001615                          SDI.3    := SDI.2+SDIB.L  ; START OF SDI INTERCONNECT #3 CONTROL BLOCK
001735                          SDI.4    := SDI.3+SDIB.L  ; START OF SDI INTERCONNECT #4 CONTROL BLOCK
002055                          BUFBEG   := SDI.4+SDIB.L  ; START OF BUFFER CONTROL BLOCKS
005255                          BUFEND   := BUFBEG+<NBCB*BUF.LL>; END OF BUFFER CONTROL BLOCKS
005255                          BUFR1.   := BUFEND        ; START OF 1ST DATA BUFFER
005672                          BUFR2.   := BUFR1.+BUF.DL; START OF 2ND DATA BUFFER
033317                          DATEND   := BUFR1.+<<NBUFR+1>*BUF.DL> ; [cho4]END OF DATA BUFFERS (INCLUDING COMPARE BUFFER)
033317                          STCACH   := DATEND        ; START OF PTE CACHE AREAS
033737                          ENCACH   := STCACH+<NSDI*MEMSZ*2> ; ALLOCATE CACHE FOR EACH SDI

000000                          ASSUME   ENCACH,LT,ALGADR ; MAKE SURE NO OVERWRITTING ECC DATA
034000                          ALGADR   := 34000         ; ANTI-LOG ADDRESS
```

```
036000                          LGADR   := ALGADR+2000  ; LOG ADDRESS
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *     CONTROLLER INTERNAL ERROR CODES                                     *
                                ; *-------------------------------------------------------------------------*

000001                          ER.PRD  := 1                    ; UNIBUS PACKET READ ERROR
000002                          ER.PWR  := 2                    ; UNIBUS PACKET WRITE ERROR
000003                          ER.RP2  := 3                    ; CONTROLLER BI READ PARITY ERROR 2
000003                          ER.WP2  := 3                    ; CONTROLLER BI WRITE PARITY ERROR 2
000003                          ER.BP1  := 3                    ; CONTROLLER BCI PARITY ERROR 1
010000                          ER.SRP  := 10000                ; CONTROLLER BI PARITY ERRORS/SHIFTED FOR LEDS
000004                          ER.RAP  := 4                    ; CONTROLLER RAM PARITY ERROR
020000                          ER.SAP  := 20000                ; CONTROLLER RAM PARITY ERROR/SHIFTED FOR LEDS
000005                          ER.ROP  := 5                    ; CONTROLLER ROM PARITY ERROR
030000                          ER.SOP  := 30000                ; CONTROLLER ROM PARITY ERROR/SHIFTED FOR LEDS
000006                          ER.RRD  := 6                    ; UNIBUS RING READ ERROR
000007                          ER.RWR  := 7                    ; UNIBUS RING WRITE ERROR
000010                          ER.INT  := 8.                   ; UNIBUS INTERRUPT MASTER FAILURE
000011                          ER.HTO  := 9.                   ; HOST ACCESS TIMEOUT ERROR
000012                          ER.NIM  := 10.                  ; HOST EXECEEDED COMMAND LIMIT
000013                          ER.MST  := 11.                  ; UNIBUS BUS MASTER FAILURE
000014                          ER.DMX  := 12.                  ; DM XFC FATAL ERROR
000015                          ER.TMO  := 13.                  ; CONTROLLER HARDWARE ERROR
000016                          ER.VCI  := 14.                  ; INVALID VIRTUAL CIRCUIT IDENTIFIER
000017                          ER.IWR  := 15.                  ; INTERRUPT WRITE ERROR ON UNIBUS
000026                          ER.MRR  := 22.                  ; MAPPING REGISTER READ ERROR
                                                                ; EITHER TIMEOUT OR PARITY WHILE READING
                                                                ; PTES FROM HOST.  See Sec 8.0 UQSSP.
                                ;ER.SUN := 23.                  ; TOO MANY SUB-UNITS ON CONTROLLER kjn
                                                                ; ER.SUN is not referenced
                                                                ; elsewhere in code. 23. reserved for
                                                                ; attempt to map when mapping not supported
                                                                ; Sec 8.0 uqssp
000144                          ER.MER  := 100.                 ; BI MASTER ERROR (must be controller specific)
000146                          ER.STP  := 102.                 ; BI STOP OCCURED
000147                          ER.BCA  := 103.                 ; BCAI PARITY TO RAM ERROR
000150                          ER.RTO  := 104.                 ; [mjt08] BI RETRY TIME OUT
000150                          BERMAX  := ER.RTO
000000                                  ASSUME BERMAX,EQ,ER.RTO
                                .PAGE
```

```
                        ; *-------------------------------------------------------------------------*
                        ; *     SYSTEM MACROS                                                       *
                        ; *-------------------------------------------------------------------------*

                        .MACRO  ASSUME  V1,TST,V2               ; ASSUME TWO VALUES MEET CONDITION
                        .if tst,<<V1>-<V2>>
                        .iff

                            ?ASSUME - VALUES V1 AND V2 ARE NOT TST

                        .endc
                        .ENDM
                        .PAGE
```



```
                        ; *-------------------------------------------------------------------------*
                        ; *     SDI LEVEL 0 AND 1 EQUATES                                           *
                        ; *-------------------------------------------------------------------------*

152000                  CONTCD  := ^H0D400      ; CONTINUE FRAME CODE
164000                  ECHOCD  := ^H0E800      ; ECHO FRAME CODE
131000                  ENDCD   := ^H0B200      ; END FRAME CODE
046400                  NENDCD  := <<-ENDCD/256.>-1>*256.; NOT END FRAME CODE (SET FOR XOR IN END.F)
025400                  FORICD  := ^H02B00      ; FORMAT ON INDEX CODE
046400                  FORSCD  := ^H04D00      ; FORMAT ON SECTOR CODE
013400                  READCD  := ^H01700      ; SELECT HEAD AND READ CODE
070400                  STRTCD  := ^H07100      ; START FRAME CODE
122400                  WRITCD  := ^H0A500      ; SELECT HEAD AND WRITE CODE
107000                  SGRPCD  := ^H08E00      ; SELECT GROUP CODE
175377                  WRRDX1  := <-<READCD&WRITCD>-1>
131000                  WRRDXR  := <<READCD!WRITCD>&WRRDX1>; READ/WRITE XOR VALUE

                        ; *-------------------------------------------------------------------------*
                        ; *     SDI LEVEL 2 EQUATES                                                 *
                        ; *-------------------------------------------------------------------------*

000176                  COMPLT  := ^H07E        ; COMPLETED OP CODE
000201                  CHGMOD  := ^H081        ; CHANGE MODE OP CODE
000202                  CHGFLG  := ^H082        ; CHANGE CONTROLLER FLAGS OP CODE
000001                  DCN.ST  := BIT00        ; DISCONNECT SPIN DOWN FLAG
000200                  DCN.TT  := BIT07        ; DISCONNECT TERMINATE TOPOLOGY FLAG
000204                  DISCON  := ^H084        ; DISCONNECT OP CODE
000005                  DRVCLR  := ^H005        ; DRIVE CLEAR OP CODE
000006                  ERECOV  := ^H006        ; ERROR RECOVERY OP CODE
000207                  GTCCHR  := ^H087        ; GET COMMON CHARACTERISTICS OP CODE
000170                  GTCRSP  := ^H078        ; GET COMMON CHARACTERISTICS RESPONSE CODE
000011                  GETSTA  := ^H009        ; GET STATUS OP CODE
000210                  GTUCHR  := ^H088        ; GET SUBUNIT CHARACTERISTICS OP CODE
000167                  GTURSP  := ^H077        ; GET SUBUNIT CHARACTERISTICS RESPONSE CODE
000012                  INSEEK  := ^H00A        ; INITIATE SEEK OP CODE
000213                  ONLINE  := ^H08B        ; ONLINE OP CODE
000014                  RUNN    := ^H00C        ; RUN OP CODE
000216                  RECALB  := ^H08E        ; RECALIBRATE OP CODE
000003                  SDIRTY  := 3            ; SDI LEVEL TWO ERROR RETRY COUNT + 1
000200                  SDTIMO  := 200          ; SERDES TIMEOUT
000006                  SEEKL   := 6            ; LENGTH OF SDI LEVEL 2 SEEK COMMAND
000366                  STARSP  := ^HOF6        ; GET STATUS RESPONSE
000220                  TOPOL   := ^H090        ; TOPOLOGY OP CODE
000157                  TOPRSP  := ^H06F        ; TOPOLOGY RESPONSE
000017                  SDITMO  := 15.          ; CONTROLLER TIMEOUT VALUE (15 SECONDS)
000175                  UNSUCC  := ^H07D        ; UNSUCCESSFUL OP CODE

                        ; *-------------------------------------------------------------------------*
                        ; *     SDI ERROR AND OTHER EQUATES                                         *
                        ; *-------------------------------------------------------------------------*

000010                  CSERR   := BIT03        ; CHECKSUM ERROR ON LEVEL 1 READ
000001                  DCERR   := BIT00        ; DRIVE CLOCK TIMEOUT ON LEVEL 1 READ/WRITE
```

```
000004                          FRERR   := BIT02        ; FRAMING ERROR ON LEVEL 1 READ
000020                          LNERR   := BIT04        ; RESPONSE LENGTH ERROR
000002                          SFERR   := BIT01        ; FIRST WORD NOT START FRAME ERROR
000040                          UNERR   := BIT05        ; UNSUCCESSFUL RESPONSE ERROR
                                .PAGE
```



```
                                .SBTTL  KDB U-PROC DIAGNOSTICS & INITIALIZATION CODE

                                ;;******************************************************************************
                                ;
                                ;       DIAGNOSTIC HISTORY
                                ;
                                ;;******************************************************************************

                                ;16-JAN-81       MATT    NPR FIX
                                ;20-JAN-81       CURT    FIX ZERO BUFFER,SAVE FAILURE
                                ;27-JAN-81       MATT    FIXED TWO STEP BITS SET IN SA REG
                                ;29-JAN-81       MATT    SDI TEST FIX
                                ;30-JAN-81       MATT    TEST CRTDS BIT IN DCR
                                ;12-FEB-81       MATT    LED CORRECTION
                                ;18-FEB-81       BILL    BURST VALUE CORRECTION
                                ;25-FEB-81       MATT    NPR ERROR REPORTING CORRECTION
                                ;27-MAR-81       MATT    IMPLEMENT RICHY'S CODE SAVING
                                ;17-JUN-81       MATT    CODE NOT TO TOUCH FAILUR WORD
                                ;18-JUN-81       MATT    CORRECTIONS OF PREVIOUS EDIT
                                ;19-JUN-81       MATT    PURGE/POLL WIPES RO FIXED
                                ;24-JUN-81       CURT    ADD SIGNITURE ANALYSIS CODE - 1 WORD
                                ;24-JUN-81       MATT    TOOK EXTRA WORD OUT OF ERROR ROUTINE/COMMENT CHANGE
                                ;30-JUN-81       BILL    REMOVE LAST FAILURE STORE
                                ;16-JUL-81       BILL    FIX CROM TIMEOUT VALUES
                                ;03-AUG-81       MATT    FIX STEP 1 DISPLAY ERROR
                                ;14-AUG-81       MATT    ACLO FIX(ONE LAST TIME)
                                ;17-NOV-81       MATT    FROM 4K TO 16K
                                ;16-FEB-82       MATT    SDI CHANGES FOR 8 PORT DIAGNOSTIC MODE
                                ;23-FEB-82       MATT    ADD '+DRINIT' AT STEP4:
                                ;05-MAY-82       MATT    TOOK OUT DIAPRT -- FOR ALL REFERENCES SEARCH FOR ;MJT(SDI)
                                ;14-MAY-82       MATT    FIXED MAX RING LEN NOT CLEARING RING AREA PROBLEM
                                ;28-MAY-82       MATT    FIXED WRAP FEATURE, LEDS FLASH ;MJT1
                                ;18-JUN-82       CURT    UDA52 NEW BOARD CHANGES
                                ;19-JUN-82       CURT    UDA52 NEW BOARD CHANGES - NOW IT WORKS
                                ;23-JUL-82       MATT    CHANGE DCRTST TO REPORT BOARD 2 ERROR
                                ;23-NOV-82       MATT    COMMENTED OUT U&D CR TESTS AS CODE REDUCTION ; [U52EC1]
                                ;04-FEB-83       MATT    SPLIT UP U AND D PROC DIAGNOSTICS
                                ;10-OCT-83       MATT    START CHANGING CODE FOR REAL QDA
                                ;18-OCT-83       MATT    CONTINUE TO CHANGE FOR REAL QDA
                                ;01-MAR-1984     MIKE    CONTINUE TO CHANGE FOR REAL UDA/QDA
                                ;11-APR-1984     MIKE    SDI PORT WRAP TEST & DFAIL U-CROM TEST
                                ;24-APR-1984     MIKE    MAKE HOST SEE SA DATA AT BEGINNING OF TEST (BL0)
                                ;04-MAY-1984     MIKE    SET QB,MP BITS IN STEP 1 & @SRUN BEFORE READING (UDS)
                                ;14-MAY-1984     MIKE    IMPLETMENTED DMA TEST TO HOST MEMORY & NOW USING PURGE
                                ;                         SIGNAL TO DETECT SA REG WRITE FROM HOST IN STEP 1.
                                ;                        PUT '.ORG 700' AT END OF DIAGNOSTIC SECTION.
                                ;30-MAY-1984     MIKE    CONDITIONED OUT 1 SECOND TIMEOUT LOOP IN UDIAG. (BL2)
                                ;03-JUL-1984     MIKE    FIXED PROBLEM IN DMA TEST (BL3)
                                ;24-AUG-1984     MIKE    ADDED LOOP-ON-TEST CODE, VERIFIED & CORRECTED ERROR CODES
                                ;11-SEP-1984     MIKE    ADDED SA ERROR CODES TO ROUTINES & ERROR CALLS
                                ;21-SEP-1984     MIKE    GENERAL ROUTINE CLEAN-UP & ADDING C$TYPE CONDITIONAL
                                ;                         FOR BDA DEVELOPMENT
                                ;24-SEP-1984     MIKE    FIX QDA MAXIMUM BURST RATE TO 16. WORDS
                                ;08-OCT-1984     MIKE    GENERAL ROUTINE CLEAN-UP & ADDED PRELIMINARY BDA CODE.
```

```
                                ;16-OCT-1984    BOB E   Add ERR03 for Alog/Log check error.
                                ;19-OCT-1984    MIKE    IMPLEMENT QDA DEFAULT BURST RATE OF 8 WORDS (4. LONGWORDS)
                                ;22-OCT-1984    MIKE    ADDED SA WRAP MODE FIX AND
                                ;                        INIT UBURST TO 2 IN SO DMA DOESN'T FAIL
                                ;30-OCT-1984    MIKE    MODIFIED SA WRAP MODE FIX
                                ;05-NOV-1984    MIKE    ADD COMMENT TO ERROR ROUTINE DESCRIPTION &
                                ;                        CHANGE LITERAL IN RAM BUFFER TEST
                                ;08-NOV-1984    MIKE    SOME COMMENT & RAM BUFFER TEST CHANGES
                                ;26-JUL-1985    MIKE    KDB uCODE FREEZE HERE...
                                ;21-AUG-1985    MATT    ADDED A LINE TO SAVE HARDWARE REVISION [mjt06]
                                ;06-SEP-1985    MIKE    CHANGE CODE FOR STOP CMD HARDWARE                                   [mr1001]
                                ;28-OCT-1985    MATT    ADDED RTOEVEN WITH BICSREN IN POLTST [mjt08] FOR HARDWARE CHANGE WITH RTO
                                ;13-NOV-1985    MIKE    CHANGE SOME KDB COMMENTS
                                ;                        & REPLACED IOC CCEF WITH CDONE and NO MERR                          [mr1002]
                                ;20-NOV-1985    MIKE    SKIP BISTOP TEST, PROBLEM CAUSED BY CCEF HARWARE CHANGE             [mr1003]
                                ;11-DEC-1985    MIKE    REMOVE BISTOP TEST                                                  [mr1004]
                                ;16-JAN-1986    MIKE    FIXED THE BI MASTER RESET TIMING PROBLEM WHEN VECTORING TO
                                ;                        ADDRESSES 7771 & 7772.                                              [mr1005]
                                ;03-DEC-1986   LESLIE    MANUFACTURING Q BUS TEST                                            [1th001]
                                ;29-JAN-1988    MATT    MOD 16 of CODVER in STEP 4
```



```
                                ;
                                ;       DIAGNOSTIC SPECIALLY USED REGISTERS
                                ;

000000                          LT400   = 0             ;0 = OLD CODE
000001                          UDA1    = 1             ;1 = NEW UDA1 ETCH

000040                          SM      = BIT05         ;STEP1 SPECIAL MODE
000100                          MP      = BIT06         ;STEP1 MAPPING SUPPORTED
000200                          OD      = BIT07         ;STEP1 ODD BYTE ADDRESSING SUPPORTED                                [mr1001]
000400                          DIAG    = BIT08         ;STEP1 ENHANCED DIAGNOSTICS SUPPORTED
000400                          DI      = BIT08         ;STEP1 ENHANCED DIAGNOSTICS SUPPORTED
001000                          QB      = BIT09         ;STEP1 22 BIT ADDRESSING SUPPORTED

000400                          SF      = BIT08         ;STEP4 SPECIAL FUNCTION BIT

001000                          BOARD2  = BIT09         ;0 = BOARD #1, 1 = BOARD #2
100000                          ERRBIT  = BIT15         ;0 = NO ERROR, 1 = ERROR

                                ;
                                ;       DIAGNOSTIC EQUATES
                                ;

000305                          ECSUML  := 305          ;ECC SUM LOWER BYTE
022000                          ECSUMH  := 22000        ;ECC SUM UPPER BYTE
176000                          ECCMSK  := 176000       ;ECC MASK

000011                          SCLR    := 11           ; xBUS - SET/RESET xBUS INFC I/O CLEAR
000011                          RCLR    := 11           ; xBUS - SET/RESET xBUS INFC I/O CLEAR
```

```
;;*****************************************************************************
;
;       Diagnostic test sequence, with respect to time and each processor
;
;       U-PROC (TEST LABEL)                 D-PROC (TEST LABEL)
;       -------                             -------
;
;       U.RSTRT:                            D.RSTRT:
;
;       SEQUENCER RELIABILITY               SEQUENCER RELIABILITY
;       SEQUENCER STACK TEST                HANG ON 1 INSTRUCTION (TSTHNG)
;       ALU TEST (ALUTST)                      HANG D-PROC (TSTHNG)
;       CONTROL REG TEST (UCRTST)                 "        (TSTHNG)
;       DFAIL TEST                          RUNS THROUGH ALL OF U and D ROM
;
;       If an error occurs here, the D-PROC will continue to run
;       thru the U or D ROM, while the U-PROC reports the error.
;
;       RELEASE D-PROC (RELES)              SEQUENCER STACK TEST
;          HANG U-PROC (RELES)              RELEASE U-PROC (RELES)
;       DISPLAY LEDS (DSLEDS)                  HANG D-PROC (RELES)
;       ROM PE TEST (UROMPE)                        "        (RELES)
;       RELEASE D-PROC (RELES)                      "        (RELES)
;          HANG U-PROC (RELES)              ALU TEST (ALUTST)
;                 "        (RELES)          DISPLAY LEDS (DSLEDS)
;                 "        (RELES)          ROM PE and TIMEOUT (DROMPE)
;                 "        (RELES)          BOARD 2 TEST (B2TST)
;                 "        (RELES)          CONTROL REG TEST (DCRTST)
;                 "        (RELES)          RAM PE TEST (DRAMPE)
;                 "        (RELES)          RELEASE U-PROC (RELES)
;       RAM PE TEST (URAMPE)                   HANG D-PROC (RELES)
;       BI FAST SELF-TEST ENABLED ?                 "        (RELES)
;       IF SO,                                      "        (RELES)
;          RELEASE D-PROC,                  BI FAST SELF-TEST ENABLED?
;          GOTO FAST TEST (FSTST)           IF SO,
;                                              GOTO FAST TEST (FSTST)
;       RELEASE D-PROC (RELES)              HANG D-PROC (RELES)
;          HANG U-PROC (RELES)              RAM BUFFER TEST (RAMTST)
;                 "        (RELES)          RELEASE U-PROC (RELES)
;       RAM BUFFER TEST (RAMTST)               HANG D-PROC (RELES)
;       RELEASE D-PROC (RELES)                      "        (RELES)
;          HANG U-PROC (RELES)              SDI TEST (SDITST)
;                 "        (RELES)          SERDES WRITE (SDWRT)
;                 "        (RELES)          SERDES READ (SDRD)
;                 "        (RELES)          ECC TEST (ECCRD)
;                 "        (RELES)          RELEASE U-PROC (RELES)
;
;                              Continue...
```



```
;
;       U-PROC (TEST LABEL)                 D-PROC (TEST LABEL)
;       -------                             -------
;
;       FSTST:                              FSTST:
;
;       SET BI LOOPBACK MODE                   HANG D-PROC (RELES)
;       BCAI BUFFER TEST (BCATST)                   "        (RELES)
;       BI PE TEST (BIPE) *PART MFG*                "        (RELES)
;       BIIC BUFFER TEST (BIITST)                   "        (RELES)
;       GET BI NODE ID (GETID)                      "        (RELES)
;       SET BI NORMAL MODE                          "        (RELES)
;       BIIC BUFFER TEST (BIITST)                   "        (RELES)
;       POLL TEST (POLTST)                          "        (RELES)
;       BI MEMORY TEST (BIMEM) *MFG*                "        (RELES)
;       LOAD BIDTYP REGISTER                        "        (RELES)
;       CLEAR BROKE BIT IN BICSR                    "        (RELES)
;       SET STEP 1 IN SA REGISTER                   "        (RELES)
;
;       U.END:                              D.END:
;
;       WAIT 50ms FOR HOST RESPONSE                 "        (RELES)
;       REDO DIAGNOSTIC ???
;       IF SO,                              IF SO,
;          SET FLAG, RELEASE D-PROC,           GOTO RESTART (D.RSTRT)
;          GOTO RESTART (U.RSTRT)
;       GOTO INITIALIZATION                 GOTO D-PROC IDLE LOOP
;       STEP.1,                             INITIALIZE DRIVES
;       STEP.2,
;       STEP.3,
;       DMA TEST (DMATST),
;       STEP.4,
;       GOTO U-PROC IDLE LOOP.
;
;;*****************************************************************************
```

```
                                ;;*****************************************************************************
                                ;
                                ; LEVO5 ** USES THE ENTIRE STACK **
                                ;
                                ;       The 2911 sequencers are tested with CONTINUES, JUMPS, CALLS and
                                ;       RETURNS to make sure they are functioning properly.  The STACK is
                                ;       tested to make sure that it's 4 locations can be wrapped around.
                                ;
                                ;       When the diagnostic is restarted, the D-PROC will be within 2 cycles
                                ;       of the U-PROC.  If it isn't, a race with the registers may occur and
                                ;       cause an error.
                                ;
                                ;       REGISTERS USED:
                                ;               Q  (TSTCNT) is used as the test counter.
                                ;               R3 (HANG) holds the SA register STEP bits until the D-PROC
                                ;                hangs.  Then, this register becomes a hand shaking mechanism
                                ;                between the U-PROC and the D-PROC, to coordinate the HANG/
                                ;                RELEASE process.
                                ;               R7 (UER) is used as the U-PROC error register.
                                ;               R17 is used as the STACK counter.
                                ;
                                ;;*****************************************************************************

000000  120457  007637  010000          INC     R17\0,,BAR @SCLR                ;WHEN RESET, INCREMENT R17 AND DISPLAY ON
                                                                                ; DATA BUS. FALL THROUGH TO RESET TIMERS
000001  013740  006310  010000          MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG     ;LOAD BREG WITH ABORT CMD, LOOPBACK MODE,
                                                                                ; BI BAD, ON POWER-UP OR WHEN SST IS ENABLED
000002  034457  004017  130005  U.RSTRT: CLR     R17,,UCRD       %CALL JMPTST    ;CLEAR UCRD, R17
                                                                                ; PUSH SEQERR ADDRESS ONTO STACK FOR
                                                                                ; POTENTIAL PARITY ERRORS.
                                ;;*****************************************************************************
                                ;
                                ;       THE SEQUENCE ERROR CODE IS EXECUTED IN AN ATTEMPT TO CONTROL A FATAL
                                ;       SEQUENCER ERROR.
                                ;
                                ; NOTE: Fatal sequencer errors, try to display 104000 somewhere.
                                ;
                                ;;*****************************************************************************

000003  013740  006310  120004  SEQERR: MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG %CALL .+1 ;LOAD BREG WITH ABORT CMD,
                                                                                ; LOOPBACK MODE & BI BAD /
                                                                                ; SEQUENCER ERROR
000004  034444  004004  100003          CLR     CRI,,UCRD       %JMP SEQERR     ;DISPLAY ALL THE LEDS /
                                                                                ; JUMP TO SEQUENCER ERROR.
```



```
000005                                  ASSUME HANG,EQ,R3

000005  034443  000603  000003  JMPTST: CLR     HANG            @RUPF %JUMPN SEQERR ;ERROR IF THIS JUMP WORKS /
                                                                                ; RESET U-PROC FLAG TO INDICATE
                                                                                ; NO LOOP ON TEST.
000006  013440  000000  030003          NOP                             %CUMPN  SEQERR  ;ERROR IF THIS CALL WORKS
000007  004240  000660  100015          CLR     TSTCNT  @RBCAI  %JMP    JUMP    ;CLEAR TSTCNT(Q) / RESET BCAI VALID BITS
                                                                                ; TRY FIRST JUMP.
000010  133750  000210  130003          MOV     #ERR00,UDDI     %CALL   SEQERR  ;ERROR JUMP DID NOT WORK
                                ;;*****************************************************************************
                                ;
                                ;       LOAD UDD (SA REG) WITH STEP DATA & RETURN ROUTINE
                                ;
                                ;;*****************************************************************************

000011  013440  000000  127777  STEP:   NOP                             %RET            ;TRY FIRST RETURN.
000012  133750  000210  100003          MOV     #ERR00,UDDI     %JMP    SEQERR  ;ERROR IF RETURN DID NOT WORK.
000013  013440  000000  000000          NOP                                             ;;To Mis-align D-PROC & U-PROC code
                                ;;*****************************************************************************
                                ;
                                ;       CROM PARITY ERROR
                                ;
                                ;;*****************************************************************************

000014  013440  000000  137777  TSCRPE: .WORD   13440,0,137777                 ;FORCE CROM PARITY ERROR
                                ;;*****************************************************************************
                                ;
                                ;       TEST RETURN
                                ;
                                ;;*****************************************************************************

000015  013440  000000  120011  JUMP:   NOP                             %CALL   STEP    ;SEE IF RETURN WORKS.
000016  034443  000003  000000          CLR     HANG                            ;RETURN WORKED
```

```
                                ;;*****************************************************************************
                                ;
                                ;       TEST STACK WRAP AROUND
                                ;
                                ;;*****************************************************************************

                                                                                ;IF WE GET HERE, BOTH JMP, CALL & RET
                                                                                ; WORKED, ELSE
                                                                                ; ALU ERROR WILL RETURN TO 'SEQERR'.
000017  033757  000004  110022  SEQTST: MOV     #4,R17          %JMP    2$      ;SET STACK PUSH COUNT
                                                                                ; LOOP HERE FOREVER IF ALU
                                                                                ; STACK OR NZRO FAILED.
000020  031477  010017  070020  1$:     DEC\T    R17             %CNZRO  1$      ;PUSH '.+1' ON STACK 3 TIMES.
000021  030557  000010  137777          ADD     #8.,R17         %RET            ;RETURN HERE 3 TIMES, THEN
                                                                                ; RETRUN TO '2$+1'.
000022  031457  000017  130020  2$:     DEC     R17             %CALL   1$      ;PUSH REAL RETURN ADDRESS
                                                                                ; STACK TEST RETURN HERE
000023  016557  010070  100003          XOR     #56.,R17\N      %JZRO   SEQERR  ;ERROR IF R17 IS ZERO /
                                                                                ; GO THRU HERE TWICE,
                                                                                ; 1st R17= 32. & 2nd R17= 56.
000024  033753  010004  037777          MOV     #4,R13          %RNZRO          ;TEST STACK WRAP-AROUND
                                                                                ; EQUALS RETURN TO '1$+1'
                                                                                ; STAY THERE 3 TIMES UNTIL STACK
                                                                                ; WRAPS AROUND.
                                                                                ; IF WE GET HERE, WRAP-AROUND WORKS
                                                                                ; WORKS, ELSE RETURN TO 'SEQERR'.
000025  031453  000013  130076  SEQ01:  DEC     R13             %CALL   PSTACK  ;PUSH STACK FULL OF ERROR
                                                                                ; RETURN ADDRESSES.
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEVOO WILL HANG HERE
                                ;
                                ;       ERROR ROUTINES
                                ;
                                ;       Red LED codes displayed;
                                ;               9 (hex) = BOARD 1 BAD
                                ;               A (hex) = BOARD 2 BAD
                                ;
                                ;       Yellow LED codes displayed;
                                ;               ON  = NODE PASSED SELF-TEST
                                ;               OFF = NODE FAILED SELF-TEST
                                ;
                                ;       Format of error code that will be put into the SA register;
                                ;
                                ;               15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
                                ;               --+--------+--------+--------+--------+--------
                                ;               E  S  S  S  S  P  B  E           T
                                ;               R  T  T  T  T  R  O     R           S
                                ;               R  E  E  E  E  O  A        R           T
                                ;               B  P  P  P  P  C  R           CODE           COUNT
                                ;               I  4  3  2  1     D
                                ;               T
                                ;                              ^  ^
                                ;                              !  !
                                ;                              !  +----- 0= Board 1 (PROC) at fault
                                ;                              !  +----- 1= Board 2 (SDI)  at fault
                                ;                              !
                                ;                              +---------- 0= D-PROC detected error
                                ;                              +---------- 1= U-PROC detected error
                                ;
                                ;;*****************************************************************************

                                        ;;BOARD 1 ERRORS - RAMPE/CROMPE/TIMEOUT RETURN ADDRESS

000026  004240  000000  000000  ERRB1A: CLR     TSTCNT                          ;ALU TEST ERROR/TSTCNT MAY HAVE GARBAGE
                                ERRB1Z:                                         ;SPECIAL LABEL FOR ALG & LG RAM TABLE ERROR
000027  033747  000040  010000  ERRB1E: MOV     #ERR01,UER                      ;CONTROL PE/ALU TEST/CR TEST ERRORS
000030  133547  000010  110045          BIS     #STEP1,UER      %JMP    ERRB1   ;SET STEP ONE TO REPORT BRD1 ERROR

                                        ;;STEP 3 OR DMA ERRORS

000031  033747  000240  130075  ERRB10: MOV     #ERR05,UER      %CALL   CLRTST  ;CLEAR TSTCNT(Q)
000032  013740  007047  135577          MOV     #TEMP2,\N,BAR   %CALL   S.LDR5  ;RESET INTERRUPT / VECTOR IN R5
000033  133547  000040  110045          BIS     #STEP3,UER      %JMP    ERRB1   ;SET STEP 3 ERROR
```

```
                                        ;;STEP 1 INIT ERROR

000034  033747  000200  120075  ERRB1L: MOV     #ERR04,UER      %CALL   CLRTST  ;STEP 1 INITIALIZE ERROR
000035  133547  000010  110045          BIS     #STEP1,UER      %JMP    ERRB1   ;SA REG DATA WAS NOT CORRECT

                                        ;;STEP 2, 3 OR 4 INIT ERROR

000036  033647  000300  120045  ERRB1Q: BIS     #ERR06,TSTCNT,UER %CALL ERRB1   ;STEP 2,3 OR 4 INITIALIZE ERROR
                                                                                ;SET RETURN ADDR IF VECTOR


                                        ;;BI TEST ERRORS

000037  114544  000002  010000  ERRB1M: BIT     #INDIAG,CRI                     ;IN DIAG MODE?
000040  033767  010144  142071          MOV\T   #ER.MER,UER      %JZRO  U.UERR  ;BI MASTER ERROR IN FUNC MODE, ELSE
000041  003740  000037  100043          MOV     #37,TSTCNT       %JMP   ERRB1B  ;BI MASTER ERROR IN DIAG MODE
000042  004240  000000  100043  ERRB1T: CLR     TSTCNT           %JMP   ERRB1B  ;BI SELF-TEST FAILED OR CMD TIMEOUT ERROR
000043  033747  000340  010000  ERRB1B: MOV     #ERR07,UER                      ;BI TEST ERROR CODE
000044  133547  000010  110045          BIS     #STEP1,UER       %JMP   ERRB1   ;SET STEP ONE TO REPORT BRD1 ERROR


                                        ;;BOARD 1 ERRORS

000045  133744  000144  100051  ERRB1:  MOV     #<LED4+LED2+DFAIL>,CRI %JMP ERRSET ;LEDS '0110' 9 (hex) 0=ON

                                        ;;BOARD 2 ERRORS

000046  033747  000100  010000  ERRB2E: MOV     #ERR02,UER                       ;RAM PE, RAM BUFFER or RAM ERROR ANY PROCESSOR
000047  133547  000012  000000          BIS     #<STEP1+BOARD2>,UER              ;SET STEP 1 &  BRD2 TO REPORT BOARD #2 ERROR.
000050  133744  000124  100051          MOV     #<LED4+LED1+DFAIL>,CRI %JMP ERRSET ;LEDS '0101' A (hex) 0=ON

                                        ;;ERRSET is the diagnostic error trigger point...

000051  133547  000204  130052  ERRSET: BIS     #ERRBIT+BIT10,UER       %CALL 1$ ;SET ERROR & U-PROC BITS IN ERROR REG /
                                                                                ; SETUP RETURN ADDR IF VECTOR.
000052  033047  000007  000000  1$:     BIS     TSTCNT,UER                      ;SET TSTCNT(0) IN ERROR REG
000053  034443  000003  000000          CLR     HANG                            ;CLEAR HANG REG SO D-PROC CAN CONTINUE.
000054  033451  000007  120243          MOV     UER,DER          %CALL  WAIT    ;SEND ERROR STATUS TO D-PROC ERROR REG /
                                                                                ; WAIT, BI SELF-TEST MAY NOT BE COMPLETE.

                                        ;;This routine is used to help the D-PROC send the error code to the SA REG

000055  013740  006310  010000  UHELP:  MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG      ;LOAD BREG WITH ABORT CMD
000056  013740  006311  000000          MOV     #<B.NRTY+B.LOOP+B.BAD+B.NABO>,BREG ;LOAD BREG WITH NO ABORT CMD
000057  033444  004004  125633          MOV     CRI,,UCRD        %CALL  UNB.RS  ;SET LEDS ON BOARD 1 / RESET BUS
```



```
000060  133544  000002  010000  UHANG:  BIS     #INDIAG,CRI                     ;SET 'INDIAG' MODE TO RETURN FROM BCAI VECTOR
000061  133547  000200  010000          BIS     #ERRBIT,UER                     ;SET ERROR BIT
000062  033740  000364  130272          MOV     #SAREG,R0        %CALL  LRGADR  ;OFFSET TO SA REGISTER (BI GPR 1) /
                                                                                ; GO LOAD BIIC REGISTER ADDRESS.
000063  033740  000004  000000          MOV     #BIWRCM,R0                      ;DO WRITE COMMAND
000064  033440  005540  000000          MOV     R0,,BCAID    @LCOM              ;LOAD COMMAND
000065  033441  005527  000000          MOV     UER,R1,BCAID @WRFST             ;SAVE UER IN R1 /
                                                                                ; WRITE ERROR TO R1 & FIRST BUFFER
000066  034447  005747  130070          CLR     UER,,BCAID @LBWR %CALL  1$      ;CLEAR UER INCASE OF VECTOR /
                                                                                ; WRITE LAST BUFFER AND DO THE WRITE
                                                                                ; SETUP VECTOR RETURN ADDRESS
000067  033447  000001  100072          MOV     R1,UER           %JMP   3$      ;RESTORE OLD UER VALUE AFTER VECTOR
000070  037140  000000  000000  1$:     XNOR    R0,R0                           ;INITIALIZE WAIT LOOP COUNT (91ms)
000071  031440  010000  010071  2$:     DEC     R0               %JNZRO 2$      ;WAIT FOR PARITY ERROR VECTOR
000072  033444  004004  000000  3$:     MOV     CRI,,UCRD                       ;RESTORE LEDS ON BOARD 1
000073  013740  006310  010000          MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG     ;LOAD BREG WITH ABORT CMD
000074  013447  000007  110072          TST     UER              %JMP   3$      ;TEST U-PROC ERROR REG /
                                                                                ; LOOP HERE FOREVER...
```

```
                                ;;*****************************************************************************
                                ;
                                ;       CLEAR TSTCNT(Q) BECAUSE IT COULD BE FILLED WITH GARBAGE AT TIME
                                ;       OF INIT OR DMA ERROR
                                ;
                                ;;*****************************************************************************

000075  004240  000000  127777  CLRTST: CLR     TSTCNT          %RET            ;TSTCNT(Q) = 0 (COULD BE GARBAGE)

                                ;;*****************************************************************************
                                ;
                                ;       PUSH STACK FOR STACK WRAP TEST
                                ;
                                ;;*****************************************************************************

000076  000057  010017  050025  PSTACK: ADD\F   R17,TSTCNT       %JNZRO  SEQ01   ;JUMP BACK UNTIL STACK FULL, ELSE
                                                                                ; INCREMENT TSTCNT(Q).
```



```
                                ;;*****************************************************************************
                                ;
                                ;       The U-PROC calls its BOARD 1 diagnostics
                                ;
                                ;;*****************************************************************************

                                U.DIAG:
000077  037140  000000  130373  T0:     XNOR    R0,R0           %CALL   ALUTST  ;INITIALIZE R0 TO -1 / DO ALU TEST
000100  013460  010000  050215          NOP\T                   %JNZRO  U.END   ;JUMP IF ERROR
000101  014467  060007  140077          CLR\T   UER,\N          %JUPF   T0      ;LOOP IF U-PROC FLAG SET

000102  133760  010020  170456  T1:     MOV\T   #<1024.*4>,R0   %CZRO   UCRTST  ;INIT R0 WITH 4K COUNT /
                                                                                ; DO CONTROL REG DATA INTEGRITY TEST
000103  013460  010000  050215          NOP\T                   %JNZRO  U.END   ;JUMP IF ERROR
000104  014467  060007  140102          CLR\T   UER,\N          %JUPF   T1      ;LOOP IF U-PROC FLAG SET

                                ;;*****************************************************************************
                                ;
                                ; THE FOLLOWING CODE IS USED TO TEST THE DFAIL FUNCTION IN THE U-PROC'S CONTROL
                                ; REGISTER AND THE CONTROL ROM PARITY ERROR FLAG. THIS CODE ALSO CHECKS FOR ANY
                                ; BAD PARITY LOCATIONS WHICH MAY EXIST IN ROM.
                                ;
                                ; DFAIL PART 1
                                ;
                                ; The D-PROC will be forced from the hang routine (TSTHNG) in D-PROC ROM and
                                ; go back when done.  The D-PROC will increment thru the D-CROM, and then go
                                ; across a known BAD PARITY location (TSCRPE), which will set the Control ROM
                                ; Parity error.  When DFAIL test PART 1 is complete, the D-PROC will execute
                                ; the instruction at SEQTST-3 in the D-PROC ROM.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106040 - DFAIL test error
                                ;       106042 - CROM parity error
                                ;
                                ;;*****************************************************************************

000105  133544  004006  010000          BIS     #<DFAIL+INDIAG>,CRI,UCRD        ;SET DFAIL BITS TO SEND D-PROC THRU THE D-CROM.
000106  131540  120003  050027          SUB\F   #3,R0           %JCPE   ERRB1E  ;ERROR(106042) IF CPE SET HERE, ELSE
                                                                                ; SET R0 COUNTER TO 7775.
000107  031440  010000  010107  1$:     DEC     R0              %JNZRO  1$      ;WAIT HERE TIL D-PROC TESTS D-CROM PE.
000110  033757  120377  010027          MOV     #377,R17        %JCPE   ERRB1E  ;ERROR(106042) IF CPE IS SET HERE,
                                                                                ; STUFF R17 WITH HIGH VALUE JUST
                                                                                ; INCASE D-PROC IS IN SEQUENCE TEST NOW.
                                                                                ; THIS WILL FORCE THE D-PROC TO
                                                                                ; STAY IN THAT LOOP.
000111  004240  120000  110027          CLR     TSTCNT          %JNCPE  ERRB1E  ;ERROR(106040) IF CPE NOT SET HERE /
                                                                                ; CLEAR TSTCNT(Q).
000112  013440  120620  010027          NOP             @RCLR   %JCPE   ERRB1E  ;ERROR(106040) IF CPE IS SET HERE, ELSE
                                                                                ; RESET I/O CLEAR.
000113  115544  004004  120455          BIC     #DFAIL,CRI\N,UCRD %CALL INCRTN  ;RELEASE D-PROC FROM DFAIL TEST /
                                                                                ; INCREMENT TSTCNT(Q) / RETURN.
```

```
                                ;;*****************************************************************************
                                ; DFAIL PART 2
                                ;
                                ; The D-PROC will be forced from the hang routine (TSTHNG) in D-PROC ROM and
                                ; go back when done.  The D-PROC will increment thru the U-CROM, and then go
                                ; across a known BAD PARITY location (TSCRPE), which will set the Control ROM
                                ; Parity error.  When DFAIL test PART 2 is complete, the D-PROC will execute
                                ; the instruction at SEQTST-1 in the D-PROC ROM.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106040 - DFAIL test error
                                ;       106042 - CROM parity error
                                ;
                                ;*****************************************************************************

000114  133740  000020  120455          MOV     #<1024.*4>,R0   %CALL   INCRTN  ;INIT R0 WITH 4K COUNT /
                                                                                ; INCREMENT TSTCNT(Q) / RETURN.
000115  033544  004100  010000          BIS     #U8K,CRI,UCRD                   ;SET U8K BIT TO SEND D-PROC THRU THE U-CROM.
000116  131540  000003  000000          SUB     #3,R0                           ;SET R0 COUNTER TO 7775.
000117  013440  000620  010000          NOP             @RCLR                   ;RESET CPE WITH I/O CLEAR,
                                                                                ; Just incase CPE was generated while
                                                                                ; Switching DFAIL test from D-CROM to U-CROM.

000120  031440  010000  010120  2$:     DEC     R0              %JNZRO  2$      ;WAIT HERE TIL D-PROC TESTS U-CROM PE.
000121  004240  120000  050027          CLR\F   TSTCNT          %JCPE   ERRB1E  ;ERROR(106042) IF CPE IS SET HERE, ELSE
                                                                                ; CLEAR TSTCNT(Q).
000122  013440  120000  110027          NOP                     %JNCPE  ERRB1E  ;ERROR(106040) IF CPE NOT SET HERE
000123  033443  120637  000027          MOV     R17,HANG @RCLR  %JCPE   ERRB1E  ;ERROR(106040) IF CPE IS SET HERE /
                                                                                ; RESET CPE WITH I/O CLEAR /
                                                                                ; SET HANG REGISTER SO WE CAN WAIT FOR
                                                                                ; THE D-PROC TO RESET HANG LATER.
000124  035544  004100  010000          BIC     #U8K,CRI,UCRD                   ;CLEAR U8K BIT TO RETURN D-PROC TO D-CROM
000125  013440  000000  000000          NOP                                     ;WASTE A CYCLE BEFORE D-PROC RESTARTS
000126  135544  004004  000000          BIC     #DFAIL,CRI,UCRD                 ;RELEASE D-PROC FROM DFAIL TEST TO START
                                                                                ; EXECUTING IT'S SEQUENCER TEST.
000127  013443  010623  010127  3$:     TST     HANG    @RCLR   %JNZRO  3$      ;WAIT HERE TIL D-PROC GETS TO 'RELES+2' /
                                                                                ; RESET CPE WITH I/O CLEAR,
                                                                                ; Just incase CPE was generated while
                                                                                ; Switching D-PROC from U-CROM to D-CROM.
                                                                                ;
000130  133744  004362  120236          MOV     #<LEDS+INDIAG>,CRI,UCRD %CALL DSLEDS  ;TURN ALL LEDS OFF /
                                                                                ; ROTATE LED DISPLAY.
000131  003760  010030  170455          MOV\T   #30,TSTCNT      %CZRO   INCRTN  ;INITIALIZE TSTCNT(Q) FOR NEXT TEST.
000132  013760  017001  160455  T2:     MOV\T   #1,\N,BAR       %CZRO   INCRTN  ;LOCATION OF RAM PE / INC TST CNT / RTN
000133  033761  013001  170471          MOV\T   #1,R1,BUF       %CZRO   UROMPE  ;CLEAR RAM PE / ROM PE TEST
000134  013460  010000  050215          NOP\T                   %JNZRO  U.END   ;JUMP IF ERROR
000135  014467  060007  140132          CLR\T   UER,\N          %JUPF   T2      ;LOOP IF U-PROC FLAG SET
                                HU.T3:
                                HU.T4:
                                HU.T5:
000136  013460  010000  170232  HU.T6:  NOP\T                   %CZRO   RELES   ;HANG U-PROC, RELEASE D-PROC TO DO BOARD 1 DIAGS
                                                                                ; AND RAM PE TEST.
```



```
                                ;;*****************************************************************************
                                ;
                                ;       The U-PROC calls its BOARD 2 diagnostics
                                ;
                                ;;*****************************************************************************

000137  013460  010000  170475          NOP\T                   %CZRO   URAMPE  ;RAM PE TEST
000140  013460  010000  050215          NOP\T                   %JNZRO  U.END   ;JUMP IF ERROR
000141  014467  060007  150136          CLR\T   UER,\N          %JUPF   HU.T6   ;LOOP IF U-PROC FLAG SET
000142  034443  100003  150145          CLR\F   HANG            %JNFTEST HU.T7  ;JUMP IF NOT BI FAST SELF-TEST, ELSE
                                                                                ; RELEASE D-PROC
000143  132444  000004  010000          NEG     CRI                             ;INDICATE BI FAST SELF-TEST TO THE D-PROC
000144  013451  000011  100153          TST     DER             %JMP    FSTST   ;JUMP TO COMPLETE BI TESTING /
                                                                                ; DID THE D-PROC HAVE AN ERROR?

000145  013460  010000  170232  HU.T7:  NOP\T                   %CZRO   RELES   ;HANG U-PROC, RELEASE D-PROC TO DO RAM BUFFER TEST.

000146  013460  010000  170505  T8:     NOP\T                   %CZRO   RAMTST  ;RAM BUFFER TEST.
000147  000660  010001  170455          ADD\T   #1,TSTCNT        %CZRO   INCRTN  ;INITIALIZE TSTCNT(Q) FOR NEXT TEST.
000150  013460  010000  050215          NOP\T                   %JNZRO  U.END   ;JUMP IF ERROR
000151  014467  060007  140146          CLR\T   UER,\N          %JUPF   T8      ;LOOP IF U-PROC FLAG SET
000152  013460  010000  170232  HU.T9:  NOP\T                   %CZRO   RELES   ;HANG U-PROC, RELEASE D-PROC TO DO REST OF
                                                                                ; BOARD 2 DIAGNOSTICS.
```

```
                                ;;*****************************************************************************
                                ;
                                ;       The U-PROC calls its BI diagnostics
                                ;
                                ;;*****************************************************************************

                                T9:
000153  013440  010000  010215  FSTST:  NOP                     %JNZRO  U.END     ;JUMP IF ERROR
000154  033746  000321  010000          MOV     #<B.NRTY+B.LOOP+B.LED+B.NABO>,R6 ;STORE BREG VALUE IN R6,
                                                                                 ; NO ABORT CMD, ASSUME THAT BIIC SELF-TEST OK,
                                                                                 ; LOOPBACK MODE & NO RETRY
000155  015546  116001  056237          BIC\F   #B.NABO,R6\N,BREG %JSTOP BI.STP ;JUMP IF BI STOPPED, ELSE                [mr1001]
                                                                                 ; ABORT BI COMMAND (RESET BIIC)
000156  033446  006006  010000          MOV     R6,,BREG                         ;LOAD BREG
000157  133745  000100  120501          MOV     #<1024.*16.>,UBAR %CALL RAMZAP   ;CLEAR 16K RAM BUFFER AFTER SFTSO
000160  013740  007266  000000          MOV     #BADRH,\N,BAR                    ;POINT TO HI WORD OF BI BASE ADRS IN RAM
000161  113740  003140  010000          MOV     #<LW+IOACC>,\N,BUF               ;STORE LONGWORD DATA LENGTH & IO ACCESS ENABLED
                                                                                 ; IN HI WORD OF BI BASE ADDRESS

                                        ;;Do BI testing in LOOPBACK mode

000162  003740  000001  120530          MOV     #1,TSTCNT       %CALL   BCATST   ;LOAD TEST COUNT /
                                                                                 ; DO BCAI REGISTER FILE TEST
000163  033760  010004  160313          MOV\T   #BICSR,R0       %CZRO   RDCMD    ;READ BI CONTROL REGISTER
000164  114561  010010  140165          BIT\T   #<STS>,R1       %TZRO            ;SELF-TEST SUCCESSFUL SET?
000165  013447  010007  150042          TST\F   UER             %JZRO   ERRBIT   ;ERROR IF STS NOT SET
000166  003760  010002  170630          MOV\T   #2,TSTCNT       %CZRO   BIPE     ;LOAD TEST COUNT /
                                                                                 ; DO BI PARITY TEST
000167  003760  010003  160751          MOV\T   #3,TSTCNT       %CZRO   BIITST   ;LOAD TEST COUNT /
                                                                                 ; DO BIIC BUFFER TEST

                                        ;;Do BI testing in NORMAL mode

000170  033766  010221  160253          MOV\T   #<B.NRTY+B.LED+B.NABO>,R6 %CZRO GETID ;STORE BREG VALUE IN R6,
                                                                                      ; LOAD BREG WITH NO RETRYS,
                                                                                      ; SELF-TEST OK & NO ABORT CMD /
                                                                                      ; GET NODE ID
000171  033466  016006  160252          MOV\T   R6,,BREG        %CZRO   TSTRTN   ;LOAD BREG
000172  003760  010004  170751          MOV\T   #4,TSTCNT       %CZRO   BIITST   ;LOAD TEST COUNT /
                                                                                 ; DO BIIC BUFFER TEST
000173  003760  010005  171014          MOV\T   #5,TSTCNT       %CZRO   POLTST   ;LOAD TEST COUNT /
                                                                                 ; DO POLL TEST
000174  100260  010000  170252          INC\T   TSTCNT          %CZRO   TSTRTN   ;LOAD TEST COUNT                        [mr1004]
000175  100260  010000  171026          INC\T   TSTCNT          %CZRO   BIMEM    ;LOAD TEST COUNT /
```



```
                                        ;;Write the uCODE VERSION & DEVICE TYPE to the BIDTYP register

000176  013460  010000  050215          NOP\T                   %JNZRO  U.END    ;JUMP IF ERROR
000177  033740  000000  120313          MOV     #BIDTYP,R0      %CALL   RDCMD    ;READ BI DEVICE TYPE REGISTER
000200  135542  000377  000000          BIC     #HIBYT,R2                        ;INIT uCODE VERSION BITS
000201  013740  007275  135621          MOV     #BUFP49,BAR     %CALL   S.STR2   ;SAVE THE HARDWARE VERSION NUMBER [mjt06]
000202  133542  000023  010000          BIS     #<CODVER>*256.,R2                ;LOAD HI BITS OF DEV REV FIELD (uCODE VERSION)
000203  033741  000016  010000          MOV     #<B.DTYP>&LOBYT,R1               ;LOAD LO BITS &
000204  133541  000001  010000          BIS     #<B.DTYP>&HIBYT,R1               ; HI BITS OF DEVICE TYPE FIELD
000205  033740  000000  120302          MOV     #BIDTYP,R0      %CALL   WRCMD    ;WRITE BI DEVICE TYPE REGISTER

                                        ;;Clear BROKE bit in BICSR register

000206  033740  000004  130313          MOV     #BICSR,R0       %CALL   RDCMD    ;READ BI CONTROL REGISTER
000207  114541  000020  010000          BIT     #BROKE,R1                        ;IS BROKE BIT SET?
000210  033760  010004  070302          MOV\T   #BICSR,R0       %CNZRO  WRCMD    ;IF SO, GO RESET BROKE BIT

                                        ;;End diagnostics if HOST WROTE to SA REG, else WRITE STEP 1 data to SA REG

000211  033740  000054  130313          MOV     #BIWSTA,R0      %CALL   RDCMD    ;READ BI WRITE STATUS REGISTER
000212  114542  000040  010000          BIT     #<GPR1>,R2                       ;WAS SA REG WRITTEN BY HOST?
000213  133741  010011  050215          MOV\F   #<STEP1+DI>,R1 %JNZRO U.END      ;JUMP IF DATA RECEIVED FROM HOST, ELSE
                                                                                 ;LOAD R1 WITH STEP 1 DATA -
                                                                                 ; (DON'T SET 22 BIT ADDR ON BI)
                                                                                 ; ENHANCED DIAGNOSTICS,
000214  033541  000140  130365          BIS     #<MP+SM>,R1     %CALL   WRTSA    ; MAPPING+SPECIAL MODE BITS &
                                                                                 ; SEND STEP DATA TO HOST SA REG IN LOOPBACK MODE
```

```
                                ;;*****************************************************************************
                                ;
                                ;       U END...
                                ;
                                ;;*****************************************************************************

000215  013451  000011  000000  U.END:  TST     DER                                     ;DID THE D-PROC HAVE AN ERROR?
000216  013447  010007  040055          TST\F    UER             %JNZRO  UHELP   ;IF SO, LET U-PROC HELP REPORT ERROR TO HOST
                                                                                ; DID THE U-PROC HAVE AN ERROR?
000217  033746  010005  040045          MOV\F    #5,R6           %JNZRO  ERRB1   ;IF SO, REPORT SOMETHING WENT WRONG WITH BD 1, ELSE
                                                                                ; INITIALIZE HOST RESPONSE WAIT COUNT.

                                ;;End diagnostics if HOST WROTE to SA REG, else RESTART DIAGNOSTICS again...

000220  033740  000054  130313  1$:     MOV      #BIWSTA,R0      %CALL   RDCMD   ;READ BI WRITE STATUS REGISTER
000221  114542  000040  010000          BIT      #<GPR1>,R2                      ;WAS SA REG WRITTEN BY HOST?
000222  034463  010003  051147          CLR\T    HANG            %JNZRO  STEP.1  ;IF SO, RELEASE D-PROC /
                                                                                ; JUMP TO STEP.1 TO START INITIALIZATION
000223  013440  000000  120243          NOP                      %CALL   WAIT    ;WAIT 10ms
000224  031446  000006  000000          DEC      R6                              ;SHOULD WE LOOK AT SA REG AGAIN?
000225  034443  010003  040220          CLR\F    HANG            %JNZRO  1$      ;JUMP IF YES, ELSE
                                                                                ; RELEASE D-PROC TO BE WITHIN 2 CYCLES
000226  013440  000000  000000          NOP                                     ; OF THE U-PROC WHEN REDOING THE DIAGNOSTICS.
000227  004240  000000  120252          CLR      TSTCNT          %CALL   TSTRTN  ;RELEASE D-PROC / WASTE AN INSTRUCTION,
000230  013440  000000  120252          NOP                      %CALL   TSTRTN  ; AN SOME MORE INSTRUCTIONS
000231  013440  000000  100002          NOP                      %JMP    U.RSTRT ; SO D-PROC CAN RESTART IN SEQUENCE WITH U-PROC.
                                                                                ; GO REDO DIAGNOSTICS...
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEV01
                                ;
                                ;       RELEASE THE D-PROC FROM HANGING AND LET U-PROC HANG.  RETURN WILL OCCUR
                                ;       WHEN THE HANG REGISTER IS CLEARED.
                                ;
                                ;       Note:   The D-PROC ERROR REGISTER (R7) will be tested on return.
                                ;
                                ;;*****************************************************************************

000232  034443  000003  120252  RELES:  CLR      HANG            %CALL   TSTRTN  ;RELEASE D-PROC AND
                                                                                ; WASTE ONE INSTRUCTION.
000233  033743  000001  000000          MOV      #1,HANG                         ;KEEP SOMETHING IN HANG
000234  013451  010011  137777  1$:     TST      DER             %RZRO           ;IF ZERO, TEST D-PROC ERROR/RETURN
000235  013443  000003  110234          TST      HANG            %JMP    1$      ;SEE IF PROC SHOULD LEAVE LOOP?

                                ;;*****************************************************************************
                                ;
                                ; LEV01
                                ;       DISPLAY LEDS
                                ;
                                ;               TO SHOW THAT THEY DO FLASH AND THAT THE CODE IS WORKING
                                ;
                                ;       LEDS     8       4       2       1
                                ;                off     off     off     on
                                ;                off     off     on      off
                                ;                off     on      off     off
                                ;                on      off     off     off
                                ;
                                ;       CALLS WAIT
                                ;
                                ;               CR HAS LEDS SET OFF & IN DIAGNOSTIC MODE
                                ;
                                ;;*****************************************************************************

000236  133742  000020  000000  DSLEDS: MOV      #<LED1>,R2                      ;START WITH LED#1
000237  013467  030007  167777  1$:     TST\T    UER             %RNEG           ;JUMP IF DONE, ELSE
000240  035144  004002  120243          BIC      R2,CRI,UCRD     %CALL   WAIT    ;LED ON
000241  033144  004002  000000          BIS      R2,CRI,UCRD                     ;LED OFF
000242  073442  000002  110237          SHF\L    R2              %JMP    1$      ;NEXT LED
```

```
                                ;;*****************************************************************************
                                ;
                                ; LEV02
                                ;       WAIT
                                ;
                                ;       WAIT APPROXIMATELY 10ms, BASED ON 346ns SEQUENCER CLOCK
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               R0 is changed.
                                ;
                                ;;*****************************************************************************

000243  133740  000161  000000  WAIT:   MOV     #70400,R0                       ;INITIALIZE LOOP TO 10.ms COUNT
000244  031440  010000  000244  1$:     DEC     R0              %JNZRO  1$      ;KEEP LOOPING TIL DONE
000245  013440  000000  127777          NOP                     %RET            ;RETURN

                                ;;*****************************************************************************
                                ;
                                ; LEV01
                                ;       WAITS
                                ;
                                ;       WAIT APPROXIMATELY 177us, BASED ON 346ns SEQUENCER CLOCK
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               R0 is changed.
                                ;
                                ;;*****************************************************************************

000246  133740  000002  010000  WAITS:  MOV     #1000,R0                        ;INITIALIZE LOOP TO 177.us COUNT
000247  031440  010000  000247  1$:     DEC     R0              %JNZRO  1$      ;KEEP LOOPING TIL DONE
000250  013440  000000  127777          NOP                     %RET            ;RETURN

                                ;;*****************************************************************************
                                ;
                                ; LEV01 OR LEV02
                                ;
                                ;       CLRERR
                                ;               CLEAR THE ERROR REG (R7)
                                ;
                                ;;*****************************************************************************

000251  034447  000007  127777  CLRERR: CLR     UER             %RET            ;CLEAR AND RETURN
```



```
                                ;;*****************************************************************************
                                ;
                                ;       TSTRTN ROUTINE
                                ;
                                ;;*****************************************************************************

000252  013447  000007  127777  TSTRTN: TST     UER             %RET            ;RETURN
                                                                                ; THIS RETURN IS USED BY OTHERS
                                                                                ; FOR THE SAME PURPOSE: SET RETURN STATUS
                                                                                ; TO SEE IF THE NEXT TEST IS EXECUTED

                                ;;*****************************************************************************
                                ;
                                ;       GET NODE ID FOR NORMAL MODE TESTING
                                ;
                                ;       Input:
                                ;           (Entered at GETID)
                                ;               none
                                ;
                                ;           (Entered at SHFID)
                                ;               R1 contains NODE ID to be shifted into the BI BASE ADDRESS
                                ;                   LOCATION in RAM.
                                ;       Output:
                                ;               RAM location BADRL will have the LO word of the BI BASE
                                ;                   ADDRESS with the shifted NODE ID value.
                                ;               RAM location BADRH will have the HI word of the BI BASE.
                                ;                   ADDRESS with the shifted NODE ID value, longword DATA
                                ;                   LENGTH & IO ACCESS enabled.
                                ;
                                ;;*****************************************************************************

000253  033740  000004  130313  GETID:  MOV     #BICSR,R0       %CALL   RDCMD   ;READ BI CONTROL REGISTER
000254  004240  000000  000000  SHFID:  CLR     Q                               ;INIT Q VALUE
000255  034541  000017  000000          AND     #NODEID,R1                      ;SAVE NODE ID
000256  033740  000015  000000          MOV     #13.,R0                         ;# OF SHIFTS FOR ID
000257  063441  010001  150263  1$:     ROT\LQF R1              %JZRO   2$      ;JUMP IF DONE SHIFTING, ELSE
                                                                                ; SHIFT Q & R1 LEFT AGAIN
000260  003660  030001  140261          BIS\T   #BIT00,Q        %TMSB           ;IF MSB OF R1 SET, THEN SET LSB OF Q
000261  035541  000001  000000          BIC     #BIT00,R1                       ;CLEAR BIT ROTATED FROM MSB TO LSB
000262  031440  000000  110257          DEC     R0              %JMP    1$      ;COUNT DOWN SHIFTED BITS /
                                                                                ; CONTINUE 'TIL DONE
000263  013740  007265  135620  2$:     MOV     #BADRL,\N,BAR     %CALL S.STR1  ;STORE LO WORD OF BIIC BASE ADRS IN RAM
000264  103640  003140  100252          BIS     #<LW+IOACC>,Q,BUF %JMP  TSTRTN  ;STORE HI WORD OF BIIC BASE ADRS IN RAM /
                                                                                ; TEST UER & RETURN.
```

```
                                ;;*****************************************************************************
                                ;
                                ;       LOAD BI REGISTER ADDRESS (WITH NODE ID)
                                ;
                                ;       Input:
                                ;               R0 contains offset to the BI REGISTER to be accessed.
                                ;               RAM location BADRL contains the LO word of the BI BASE ADDRESS
                                ;                   (includes NODE ID if NORMAL mode).
                                ;               RAM location BADRH contains the HI word of the BI BASE ADDRESS
                                ;                   (includes NODE ID if NORMAL mode).
                                ;       Output:
                                ;               none
                                ;
                                ;;*****************************************************************************

000265  013740  007265  000000  BRGADR: MOV     #BADRL,\N,BAR                   ;POINT TO LO WORD OF BI BASE ADRS IN RAM
000266  033500  000003  010000          BIS     (BUF),R0                        ;GET LO WORD OF ADRS WITH NODE ID /
000267  033440  005560  010000          MOV     R0,,BCAID       @LADD           ;LOAD LO WORD OF ADDRESS
000270  013740  007266  000000          MOV     #BADRH,\N,BAR                   ;POINT TO HI WORD OF BI BASE ADRS IN RAM
000271  013700  005763  127777          MOV     (BUF),\N,BCAID  @HADD   %RET    ;GET HI WORD OF ADRS WITH NODE ID, DATA
                                                                                ; LENGTH & IO SPACE ACCESS CODE /
                                                                                ; LOAD HI WORD OF ADDRESS / RETURN

                                ;;*****************************************************************************
                                ;
                                ;       LOAD BI REGISTER ADDRESS (without NODE ID for error routine)
                                ;
                                ;       Input:
                                ;               R0 contains offset to the BI REGISTER to be accessed.
                                ;       Output:
                                ;               none
                                ;
                                ;;*****************************************************************************

000272  033440  005560  010000  LRGADR: MOV     R0,,BCAID       @LADD           ;LOAD LO WORD OF ADRS.
000273  133740  000140  000000          MOV     #<LW+IOACC>,R0                  ;GET DATA LENGTH & IO SPACE ACCESS CODE
000274  033440  005760  127777          MOV     R0,,BCAID       @HADD   %RET    ;LOAD HI WORD OF ADRS / RETURN

                                ;;*****************************************************************************
                                ;
                                ;       WASTE TIME ROUTINE
                                ;
                                ;;*****************************************************************************

000275  013440  000000  000000  WASTE:  NOP                                     ;WASTE SOME TIME
000276  013440  000000  000000          NOP                                     ;WASTE SOME TIME
000277  013440  000000  000000          NOP                                     ;WASTE SOME TIME
000300  013440  000000  000000          NOP                                     ;WASTE SOME TIME
000301  013440  000000  127777          NOP                     %RET            ;WASTE SOME TIME / RETURN
```



```
                                ;;*****************************************************************************
                                ;
                                ;       SUBROUTINE WRCMD
                                ;
                                ;       WRCMD  will issue a write command in BI LOOPBACK or NORMAL mode
                                ;
                                ;       Input:
                                ;               R0 contains BI WRITE command literal.
                                ;               R1 contains data to be loaded into the LO word of a BI REGISTER.
                                ;               R2 contains data to be loaded into the HI word of a BI REGISTER.
                                ;       Output:
                                ;               RAM location RGDATL will get data from R1.
                                ;               RAM location RGDATH will get data from R2.
                                ;               BREG will have LOOPBACK enabled if this routine was entered
                                ;                   at location WRLCMD.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;
                                ;;*****************************************************************************

000302  013440  000000  130265  WRCMD:  NOP                             %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS
000303  033740  000004  000000          MOV     #BIWRCM,R0                              ;GET WRITE COMMAND
000304  033440  005540  000000          MOV     R0,,BCAID   @LCOM                       ;LOAD COMMAND
000305  013740  007271  000000          MOV     #RGDATL,\N,BAR                          ;POINT TO LO REG DATA LOC IN RAM
000306  033441  003001  135621          MOV     R1,,BUF         %CALL   S.STR2  ;STORE REGISTER DATA
000307  013740  007271  000000          MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
000310  013700  005523  000000          MOV     (BUF),BCAID @WRFST              ;WRITE FIRST BUFFER
000311  013740  117272  006237          MOV     #RGDATH,\N,BAR  %JSTOP  BI.STP  ;JUMP IF BI STOPPED /                   [mr1001]
                                                                                ; POINT TO HI REG DATA LOC IN RAM
000312  013700  005743  110323          MOV     (BUF),BCAID @LBWR %JMP  WTCMDD  ;WRITE LAST BUFFER AND DO THE WRITE /
                                                                                ; GO WAIT FOR CMD COMPLETE & ERROR FREE /
                                                                                ; RETURN
```

```
                                ;;******************************************************************************
                                ;
                                ;       SUBROUTINE RDCMD
                                ;
                                ;       RDCMD will issue a BI READ COMMAND in BI LOOPBACK or NORMAL mode.
                                ;
                                ;       Input:
                                ;               R0 contains read command literal.
                                ;       Output:
                                ;               R1 will have data of 1st buffer read.
                                ;               R2 will have data of NXT buffer read.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;
                                ;;******************************************************************************

000313  013440  000000  130265  RDCMD:  NOP                     %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS
000314  033740  000001  000000          MOV     #BIRDCM,R0                      ;GET READ COMMAND
000315  033440  115540  006237          MOV     R0,,BCAID @LCOM %JSTOP  BI.STP  ;JUMP IF BI STOPPED / LOAD COMMAND      [mr1001]
000316  013440  000420  130323          NOP             @GORD %CALL   WTCMDD    ;DO THE READ /
                                                                                ; GO WAIT FOR CMD COMPLETE & ERROR FREE
000317  013740  007271  000000          MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
000320  013440  000500  000000          NOP                     @RDFST          ;READ 1ST BUFFER
000321  033701  003715  010000          MOV     (BCAIS),R1,BUF @RDNXT           ;GET DATA FROM 1ST BCAI BUFFER /
                                                                                ; READ NEXT BUFFER.
000322  033702  003015  127777          MOV     (BCAIS),R2,BUF %RET             ;GET DATA FROM NEXT BCAI BUFFER /
                                                                                ; RETURN.
```



```
                                ;;******************************************************************************
                                ;
                                ;       WAIT FOR COMMAND COMPLETE AND ERROR FREE
                                ;
                                ;       This routine will  wait approximately 10. seconds  before flagging a BI
                                ;       COMMAND TIMEOUT error.
                                ;
                                ;       Upon entering this routine,  MERR is checked  before  issuing a BI READ
                                ;       or WRITE COMMAND.   If MERR is set,  the command  is never started.  If
                                ;       MERR is clear,  wait for NCDONE (Command Not Done), which indicates the
                                ;       BI COMMAND has started.  If this does not occur within a certain amount
                                ;       of time,  then retry the command once.  If NCDONE is not true after one
                                ;       retry,  then  a BI CMD TIMEOUT error is reported.  When NCDONE is true,
                                ;       then wait for  CDONE (Command Done)  to be asserted.   If  CDONE is not
                                ;       detected within 10.sec,  a BI CMD TIMEOUT error  is reported.  If CDONE
                                ;       is detected,  the  routine  checks MERR (Master Error) and exits.
                                ;
                                ;       Input:
                                ;               R0 contains the BI COMMAND literal (BIRDCM or BIWRCM).
                                ;       Output:
                                ;               R1 & R2 will be destroyed.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;
                                ;;******************************************************************************

000323  033741  140200  000037  WTCMDD: MOV     #128.,R1        %JMERR  ERRB1M  ;ERROR IF MASTER ERROR /                [mr1002]
                                                                                ; SETUP TIMEOUT COUNT TO BE SHORT
000324  033742  160002  100337          MOV     #2,R2           %JNCDONE 4$     ;JUMP IF BI COMMAND STARTED /           [mr1002]
                                                                                ; SETUP COMMAND RETRY COUNT
                                        ;Start BI START COMMAND timeout
000325  031441  160001  140337  1$:     DEC\F   R1              %JNCDONE 4$     ;JUMP IF BI COMMAND STARTED /           [mr1002]
                                                                                ; DONE WITH CMD START WAIT LOOP YET?
000326  031442  010002  050325          DEC\F   R2              %JNZRO  1$      ;JUMP IF NOT, ELSE
                                                                                ; LAST COMMAND RETRY ?
000327  114544  010002  010331          BIT     #INDIAG,CRI     %JNZRO  2$      ;RETRY COMMAND IF NOT, ELSE
                                                                                ; ARE WE IN DIAG MODE?
000330  033742  010002  050042          MOV\F   #2,R2           %JNZRO  ERRB1T  ;DIAG MODE CMD START TIMEOUT ERROR, ELSE
                                                                                ; SETUP COMMAND RETRY COUNT
000331  112540  000004  000000  2$:     CMP     #BIWRCM,R0                      ;WAS IT A BI WRITE COMMAND ?
000332  033741  010200  010334          MOV     #128.,R1        %JNZRO  3$      ;JUMP IF NOT /
                                                                                ; SETUP TIMEOUT COUNT TO BE SHORT
000333  013440  000640  110325          NOP             @GOWR %JMP      1$      ;RETRY BI WRITE CMD

000334  112540  000001  000000  3$:     CMP     #BIRDCM,R0                              ;WAS IT A BI READ COMMAND ?
000335  013760  016310  050334          MOV\T   #<B.NRTY+B.LOOP+B.BAD>,BREG %JNZRO 3$ ;HANG IF NOT (COMMAND?)
                                                                                        ; LOAD BREG WITH ABORT CMD
000336  013440  000420  100325          NOP             @GORD %JMP      1$      ;RETRY BI READ CMD

                                        ;Start BI COMMAND COMPLETE 10 second timeout
000337  133742  160002  000346  4$:     MOV     #1000,R2        %JCDONE 6$      ;JUMP IF CMD DONE /                     [mr1002]
```

```
                                                                ; SETUP CMD CMPL OUTER TIMEOUT COUNT
000340  031441  160001  050346  5$:   DEC\F   R1        %JCDONE 6$      ;JUMP IF CMD DONE /                    [mr1002]
                                                                ; DONE WITH CMD CMPL INNER WAIT LOOP YET?
000341  031442  010002  050340        DEC\F   R2        %JNZRO  5$      ;JUMP IF NOT, ELSE DEC OUTER COUNT /   [mr1004]
                                                                ; DONE WITH CMD CMPL OUTER WAIT LOOP YET?
000342  133741  010161  010340        MOV     #70400,R1 %JNZRO  5$      ;JUMP IF NOT /
                                                                ; SETUP INNER LOOP AGAIN (10ms)
000343  114544  000002  010000        BIT     #INDIAG,CRI             ;ARE WE IN DIAG MODE?
000344  013440  010000  110337        NOP               %JZRO   4$      ;KEEP WAITING IF NOT DIAG MODE
000345  013440  000000  110042        NOP               %JMP    ERRB1T  ;ERROR IF CMD TIME OUT
000346  013440  140000  010037  6$:   NOP               %JMERR  ERRB1M  ;ERROR IF MASTER ERROR                 [mr1002]
000347  013440  000000  127777        NOP               %RET            ;RETURN                                [mr1002]
```



```
                        ;;*****************************************************************************
                        ;
                        ;       RESET BIIC BUS ERROR REGISTER
                        ;
                        ;       Input:
                        ;               none
                        ;       Output:
                        ;               none
                        ;
                        ;;*****************************************************************************
000350  137742  000200  010000  RS.BER: MOV     #77777,R2                ;GET HI &
000351  033741  000007  010000          MOV     #7,R1                    ; LO WORDS FOR BUS ERROR REG ERROR MASK
000352  033740  000010  100302          MOV     #BIBER,R0       %JMP WRCMD ;WRITE BI BUS ERROR REGISTER
                                                                ; RETURN FROM WRITE COMMAND ROUTINE.
```

```
                                ;;*****************************************************************************
                                ;
                                ;       CHECK BI PARITY
                                ;
                                ;       Input:
                                ;               R0 contains offset of BIIC register to access.
                                ;       Output:
                                ;               Test result of R2.
                                ;
                                ;;*****************************************************************************

000353  034442  000002  000000  CHKPAR: CLR     R2                              ;INIT BIT SET COUNTER
000354  013740  007265  000000          MOV     #BADRL,\N,BAR                   ;POINT TO LO ORDER BASE ADDR IN RAM
000355  033701  000003  010000          MOV     (BUF),R1                        ;GET LO ADDRESS WORD W/ NODE ID
000356  033141  000000  000000          BIS     R0,R1                           ;GET OFFSET OF REGISTER
000357  013740  007266  000000          MOV     #BADRH,\N,BAR                   ;POINT TO HI ORDER BASE ADDR IN RAM
000360  036501  000003  000000          XOR     (BUF),R1                        ;GET HI ADDRESS WORD W/ NODE ID /
                                                                                ; IS LSB BIT SET IN LO WORD?
000361  130462  040002  040362  1$:     INC\T   R2              %TLSB           ;INCREMENT BIT SET COUNT IF LSB SET
000362  053441  000001  010000          ROT\R   R1                              ;DONE ALL BITS IN WORD YET?
000363  135541  010200  010361          BIC     #BIT15,R1       %JNZRO  1$      ;CHECK NEXT BIT POSITION IF NOT DONE /
                                                                                ; IS LSB BIT SET IN LO WORD?
000364  130442  000002  127777          INC     R2              %RET            ;INC R2 TO ACCOUNT FOR BI RD OR WRT CMD /
                                                                                ; ARE ODD NUMBER OF BITS SET? /
                                                                                ; RETURN

                                ;;*****************************************************************************
                                ;
                                ;       WRITE SA REG IN LOOPBACK MODE or WRITE SA REG IN CURRENT MODE
                                ;
                                ;       This routine takes the data from R1 and writes it to the LO WORD of the
                                ;       SA REGISTER (BIGPR1).  R2 will be ZEROED and written to the HI WORD of
                                ;       the SA REGISTER.  BI BUS xfers will be executed in LOOPBACK mode to
                                ;       avoid setting the GPR1 status bit in the BIWSTA register.
                                ;
                                ;       Input:
                                ;               R1 contains DATA for the LO WORD of the SA REGISTER.
                                ;       Output:
                                ;               BI BUS will be in LOOPBACK mode.
                                ;               R2 will have ZEROS.
                                ;
                                ;;*****************************************************************************

000365  013740  006321  100367  WRTSA:  MOV     #<B.NRTY+B.LOOP+B.LED+B.NABO>,BREG %JMP  .+2 ;LOAD BREG TO WRITE TO SA REG
000366  013740  006311  000000  WRTERR: MOV     #<B.NRTY+B.LOOP+B.BAD+B.NABO>,BREG          ;LOAD BREG TO WRITE ERROR TO
                                                                                            ; SA REGISTER
000367  033740  000364  000000          MOV     #SAREG,R0                       ;OFFSET TO SA REGISTER (BI GPR 1)
000370  033440  005560  010000          MOV     R0,,BCAID @LADD                 ;LOAD LO WORD OF ADRS.
000371  034442  005762  010000          CLR     R2,,BCAID @HADD                 ;INIT HI WORD OF GPR 1
000372  033740  000004  100304          MOV     #BIWRCM,R0      %JMP    WRCMD+2 ;GO DO WRITE COMMAND
```



```
                                ;;*****************************************************************************
                                ;
                                ; TEST DESCRIPTION:
                                ;       The 2901 routine tests the internal 2901 registers, ALU functions,
                                ;       SHIFT/ROTATE functions and ALL ALU associated flags (ZERO, CARRY,
                                ;       NEGATIVE, LSB, OVERFLOW).
                                ;
                                ; Input:
                                ;       XNOR    R0,R0                           ;R0 = FFFF NEG/LSB SET
                                ;
                                ; Output:
                                ;       All registers are used in verifying register integrity &
                                ;       some registers are used in the ALU functionality.
                                ;
                                ; If any ERROR is detected within the 2901 test, the processor will report
                                ; immediately and stop any further execution.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106040 - ALU test error
                                ;
                                ;;*****************************************************************************

000373  136540  040200  140026  ALUTST: XOR\F   #BIT15,R0       %JNLSB  ERRB1A  ;R0 = 7FFF NEG CLEARED
000374  035540  030001  140026          BIC\F   #BIT00,R0       %JNEG   ERRB1A  ;R0 = 7FFE LSB CLEARED
000375  004240  040000  050026          CLR\F   Q               %JLSB   ERRB1A  ;Q = 0 ZERO SET
000376  033441  010000  050026          MOV\F   R0,R1           %JNZRO  ERRB1A  ;R1 = 7FFE ZERO CLEARED
000377  020140  017001  140026          ADD\F   R1\0,R0,BAR     %JZRO   ERRB1A  ;AD = 2 / R0 = FFFC / NEG SET
000400  100240  030000  040026          INC\F   Q               %JNNEG  ERRB1A  ;Q = 1 CARRY CLEARED
000401  102240  020000  150026          NEG\F   Q               %JCRY   ERRB1A  ;Q = FFFF LSB SET
000402  000041  040001  140026          ADD\F   R1\0,Q          %JNLSB  ERRB1A  ;Q = 7FFD CARRY SET
000403  007240  020000  050026          COM\F   Q               %JNCRY  ERRB1A  ;Q = 8002 NEG SET
000404  004260  030000  050405          CLR\T   Q               %TNNEG          ;DO NOT EXECUTE AN INSTRUCTION /
                                                                                ; SIGNAL CHANGE?
000405  101640  030002  050026          SUB\F   #2,Q            %JNNEG  ERRB1A  ;Q = 8000 LSB CLEARED
000406  033240  040000  040026          MOV\F   Q,R0            %JLSB   ERRB1A  ;R0 = 8000 ZERO CLEARED
000407  116540  010200  150026          XOR\F   #BIT15,R0\N     %JZRO   ERRB1A  ;R0 = BIT15 ZERO SET
000410  033741  010017  050026          MOV\F   #15.,R1         %JNZRO  ERRB1A  ;COUNTER R1 = 15.
000411  043440  010000  150413  1$:     SHF\RQF R0              %JZRO   2$      ;SHIFT RIGHT WITH Q, R0...IF ZERO FLAG
                                                                                ;SET HERE...GO CHECK R0 AND Q FOR SWAP
000412  031441  000001  100411          DEC     R1              %JMP    1$      ;IF NOT DONE DECREMENT SHIFT COUNTER
000413  013440  000000  000000  2$:     TST     R0                              ;IS LSB SET IN R0?
000414  013240  040000  140026          TST\F   Q               %JNLSB  ERRB1A  ;ERROR IF NOT
                                                                                ;IS LSB SET IN Q?
000415  043440  040000  140026          SHF\RQF R0              %JNLSB  ERRB1A  ;ERROR IF NOT, ELSE
                                                                                ; SHIFT R0 & Q RIGHT ONCE MORE
                                                                                ;SET HERE...GO CHECK R0 AND Q FOR SWAP
000416  013440  000000  000000          TST     R0                              ;IS MSB SET IN R0?
000417  114640  030200  040026          BIT\F   #BIT15,Q        %JNMSB  ERRB1A  ;ERROR IF NOT
                                                                                ; IS MSB SET IN Q?
000420  003740  030017  140026          MOV\F   #15.,Q          %JMSB   ERRB1A  ;ERROR IF YES
```

```
                                ;;*****************************************************************************
                                ;
                                ;       If Q NOT 0, then hang otherwise load Q with 15., for checking rest of
                                ;       registers.  Loop to check all registers by walking a bit from carry to
                                ;       carry with a register and writing and comparing it with the rest of
                                ;       the registers.  Any stuck bits will propagate and be detected.
                                ;
                                ;       R0 should be 100000 now
                                ;       This loop should start with R0 equal to 100000
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106040 - ALU test error
                                ;
                                ;;*****************************************************************************

000421  073440  000000  000000  SFT20:  ROT\L    R0                                      ;ROTATE R0 LEFT WITH BIT 15->BIT 0
000422  033441  000000  000000          MOV      R0,R1                                   ;R1=R0
000423  033442  000001  010000          MOV      R1,R2                                   ;R2=R1
000424  033443  000002  000000          MOV      R2,R3                                   ;R3=R2
000425  033444  000003  000000          MOV      R3,R4                                   ;R4=R3
000426  033445  000004  000000          MOV      R4,R5                                   ;R5=R4
000427  033446  000005  010000          MOV      R5,R6                                   ;R6=R5
000430  033457  000006  010000          MOV      R6,R17                                  ;R17=R6
000431  033450  000017  000000          MOV      R17,R10                                 ;R10=R17
000432  033451  000010  000000          MOV      R10,R11                                 ;R11=R10
000433  033452  000011  010000          MOV      R11,R12                                 ;R12=R11
000434  033453  000012  000000          MOV      R12,R13                                 ;R13=R12
000435  033454  000013  000000          MOV      R13,R14                                 ;R14=R13
000436  033455  000014  000000          MOV      R14,R15                                 ;R15=R14
000437  033456  000015  010000          MOV      R15,R16                                 ;R16=R15
000440  033447  000016  010000          MOV      R16,R7                                  ;R17=R16...LINKS ALL REGISTERS
000441  036147  000000  000000          XOR      R0,R7                                   ;HERE COMPARE R0 (IMAGE) WITH LAST REGISTER
                                                                                        ;SUPPOSEDLY CONTAINING THE SAME INFO AS R0
000442  001240  010000  050026          DEC\F    Q               %JNZRO  ERRB1A          ;ERROR IF NOT EQUAL, ELSE
                                                                                        ; DECREMENT SHIFT COUNTER.
000443  053440  010000  040421          ROT\RF   R0              %JNZRO  SFT20           ;LOOP IF Q NOT ZERO, ELSE
                                                                                        ; CHECK TO SEE NEG ISN'T SET.
000444  133744  030362  100026          MOV      #<LEDS+INDIAG>,CRI %JNEG ERRB1A        ;ERROR IF STILL NEG AFTER RIGHT ROTATE /
                                                                                        ; ALL LEDS OFF & SET DIAG MODE.
```



```
                                ;;*****************************************************************************
                                ;
                                ;       FORCE THE PAR (pipeline address register) INTO THE VECTOR TABLE
                                ;
                                ;       R7 (UER) IS PRE-LOADED WITH THE RAM PARITY ERROR SHIFTED VALUE.  ON
                                ;       RETURN IT SHOULD EQUAL THE RAM PARITY ERROR.
                                ;       PAR WILL HOLD THE ADDRESS IN THE VECTOR TABLE WHERE THE PROC WILL
                                ;       GO TO.
                                ;       CRI IS TESTED IN THE VECTOR.  OPM BIT HAS TO BE CLEARED.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106040 - ALU test error
                                ;
                                ;;*****************************************************************************

000445  173747  000040  120447  PARTST: MOV\L    #ER.SAP,R7      %CALL   1$      ;LOAD R7 VALUE / SET RETURN IN STACK
000446  036547  000004  100450          XOR      #ER.RAP,R7      %JMP    2$      ;TEST RESULTS OF R7 DURING PAR VECTOR.

000374                          HI.OFF  = HIPAR - 7400
000447  013740  002374  100027  1$:     MOV      #HI.OFF,\N,PAR  %JMP    ERRB1E  ;LOAD PAR OFFSET TO VECTOR TO HI ADDRESS /
                                                                                        ; ERROR IF PAR NOT FORCED TO HI VECTOR ADR.
000450                                  ASSUME HI.OFF,EQ,374

000450  013440  010000  010027  2$:     NOP                      %JNZRO  ERRB1E  ;ERROR IF INCORRECT RESULTS DURING
                                                                                        ; VECTOR ROUTINES
000451  173747  000040  120453          MOV\L    #ER.SAP,R7      %CALL   3$      ;LOAD R7 VALUE / SET RETURN IN STACK
000452  036547  000004  110454          XOR      #ER.RAP,R7      %JMP    4$      ;TEST RESULTS OF R7 DURING PAR VECTOR.

000003                          LO.OFF  = LOPAR - 7400
000453  013740  002003  100027  3$:     MOV      #LO.OFF,\N,PAR  %JMP    ERRB1E  ;LOAD PAR OFFSET TO VECTOR TO LO ADDRESS /
                                                                                        ; ERROR IF PAR NOT FORCED TO LO VECTOR ADR.
000454                                  ASSUME LO.OFF,EQ,3

000454  013440  010000  010027  4$:     NOP                      %JNZRO  ERRB1E  ;ERROR IF INCORRECT RESULTS DURING
                                                                                        ; VECTOR ROUTINES

                                ;;*****************************************************************************
                                ;
                                ;       INCREMENT TEST (TSTCNT= Q) AND RETURN
                                ;
                                ;;*****************************************************************************

000455  100240  000000  110252  INCRTN: INC      TSTCNT          %JMP    TSTRTN  ;INCREMENT TSTCNT(Q) & RETURN
```

```
                                ;;******************************************************************************
                                ;
                                ; LEV01
                                ;       CONTROL REGISTER DATA INTEGRITY TEST
                                ;
                                ;       Loop data around the U-PROC Control Register.
                                ;
                                ;       Don't test the DFAIL bit, it has already been tested and testing the
                                ;       DFAIL bit would be FATAL at this time.  Setting the DFAIL bit would
                                ;       cause the D-PROC to start LOOPING thru it's CROM.
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               none
                                ;
                                ;       REGISTERS USED:
                                ;               R1 is a temporary register.
                                ;
                                ;       This routine starts at bit position BIT08,
                                ;       skips over bit position BIT10 (DFAIL)
                                ;       and ends at bit position BIT15 (LED8).
                                ;
                                ;       The lo byte of this register is WRITE ONLY, so the routine will
                                ;       always ignore the lo byte of the control register.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106041 - Control register test error
                                ;
                                ;;******************************************************************************

000456  133741  000001  000000  UCRTST: MOV     #BIT08,R1                       ;LOAD R1 WITH STARTING POINT
000457  033441  004001  000000  1$:     MOV     R1,,UCRD                        ;LOAD CONTROL REGISTER
000460  033702  000014  010000          MOV     (UCRS),R2                       ;GET DATA FROM CONTROL REG
000461  035542  000377  010000          BIC     #377,R2                         ;IGNORE LO BYTE
000462  016141  000002  000000          XOR     R2,R1\N                         ;HAS DATA BEEN DESTROYED?
000463  073441  010001  040027          ROT\LF  R1              %JNZRO  ERRB1E  ;ERROR IF SO, ELSE SHIFT DATA
000464  116541  030004  150455          XOR\F   #DFAIL,R1\N     %JMSB   INCRTN  ;EXIT IF LAST BIT TESTED, ELSE
                                                                                ; IS THIS THE DFAIL BIT?
000465  073461  010001  150466          ROT\LT  R1              %TZRO           ;ROTATE LEFT IF YES, ELSE
                                                                                ; CONTINUE.
000466  013440  000000  110457          NOP                     %JMP    1$      ;DO NEXT BIT POSITION
```



```
                                ;;******************************************************************************
                                ;
                                ; TEST ROM PARITY ERROR
                                ;
                                ;       Input:
                                ;               STACK set for return.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106072 - ROM parity test error
                                ;
                                ;;******************************************************************************

000467  013440  000000  120014  TSBRPE: NOP                     %CALL   TSCRPE  ;A CONTROL ROM PARITY ERROR/SET U-PROC
000470  013440  000000  110027          NOP                     %JMP    ERRB1E  ;ERROR IF IT DOESN'T VECTOR,
                                                                                ; SHOULD VECTOR BEFORE JUMPING TO ERROR ROUTINE.

                                ;;******************************************************************************
                                ;
                                ; LEV01
                                ;       U-PROC RAM PE, CONTROL ROM PE TEST
                                ;
                                ;       VECTOR UP IN CROM MEMORY WHEN EACH ERROR OCCURS.
                                ;       SERVICE THE ERROR THEN CHECK TO MAKE SURE THE
                                ;       PROPER ERROR CODE WAS RECORDED.
                                ;
                                ;       REGISTERS USED:
                                ;       R7 (UER) HOLDS THE VECTOR ERROR CODE
                                ;
                                ;       CROM PE
                                ;
                                ;       CALLS   TSCRPE -> A KNOWN BAD PARITY LOCATION.
                                ;               TSTRTN - ONE INSTRUCTION (RET) FOR TIMING AND SETTING THE STACK
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106072 - ROM parity test error
                                ;
                                ;;******************************************************************************

000471  013440  000000  120467  UROMPE: NOP                     %CALL   TSBRPE  ;A CONTROL ROM PARITY ERROR
000472  036547  000005  010000          XOR     #ER.ROP,R7                      ;TEST RESULTS OF R7 DURING ROM PARITY VECTOR.
000473  133744  014362  000027          MOV     #<LEDS+INDIAG>,CRI,UCRD %JNZRO ERRB1E ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES / ALL LEDS OFF+DIAG MODE.
000474  013440  000000  100455          NOP                     %JMP    INCRTN  ;JUMP TO INCREMENT TEST NUMBER & RETURN.

                                ;;******************************************************************************
                                ;
                                ;       END OF ALL BOARD 1 TESTING
                                ;
                                ;;******************************************************************************
```

```
                                ;;******************************************************************************
                                ;
                                ; LEVO1
                                ;       RAM PE
                                ;
                                ;       USES RO, R7 (UER), BAR & BUF LOCATION 0
                                ;       D-PROC MUST ENABLE PARITY ERROR
                                ;
                                ;       RO IS A TEMPORARY REG
                                ;
                                ;       VECTORS INTO HIGH CROM MEMORY WHEN BAR = 0.  THIS CONDITION IS SET
                                ;               BY THE D-PROC WHEN IT RAN ITS RAM PARITY TEST
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       107103 - RAM parity error
                                ;
                                ;;******************************************************************************

000475  014440  007000  120476  URAMPE: CLR     ,\N,BAR         %CALL   1$      ;SET STACK FOR RETURN /
                                                                                ; CLEAR BAD PARITY LOCATION THAT
                                                                                ; WAS SET BY D-PROC.
000476  036547  000004  000000  1$:     XOR     #ER.RAP,R7                      ;TEST RESULTS OF R7 DURING RAM PARITY VECTOR
000477  013440  010000  000046          NOP                     %JNZRO  ERRB2E  ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES.
000500  035544  004100  110455          BIC     #U8K,CRI,UCRD   %JMP    INCRTN  ;MAKE SURE U8K BIT IS CLEAR IN CRI
                                                                                ; RESET LEDS & REENABLE SIGNALS /
                                                                                ; INCREMENT TSTCNT(Q) & RETURN.

                                ;;******************************************************************************
                                ;
                                ;       ROUTINE TO SAVE LAST FAILURE AND ZAP CONTROLLER RAM
                                ;
                                ;;******************************************************************************

000501  031445  007005  010000  RAMZAP: DEC     UBAR,,BAR                       ;LOAD RAM ADDRESS /
                                                                                ; DONE ZEROING RAM?
000502  016545  030034  127777          XOR     #FAILUR,UBAR\N  %RNEG           ;RETURN IF DONE /
                                                                                ; IS UBAR (R5) = FAILUR?
000503  034442  010002  175575          CLR\F   R2              %CZRO   S.LDR2  ;SAVE BUFFER DATA IN R2, ELSE
                                                                                ; CLEAR R2.
000504  033442  003002  110501          MOV     R2,,BUF         %JMP    RAMZAP  ;LOAD R2 INTO BUFFER / NEXT RAM LOCATION
```



```
                                ;;******************************************************************************
                                ;
                                ;LEVO1
                                ;       RAM BUFFER TEST (Duration 3.3 sec)
                                ;
                                ;       Input:
                                ;               NONE
                                ;       Output:
                                ;               Each location in RAM will be ZERO
                                ;
                                ;       REGISTERS USED:
                                ;               RO has OLD buffer value
                                ;               R1 has NEW buffer value
                                ;               R5(UBAR) has BAR value
                                ;               R6 is the increment value (1 or -1)
                                ;               R10 saves the starting value
                                ;
                                ;       TEST DESCRIPTION:
                                ;
                                ;             The Algorithm;
                                ;
                                ;             1.  Fill the RAM with all zeros.
                                ;
                                ;             2.  Starting with the lowest address,  read  a  location,
                                ;                 verifying the previous write, write a one in a single
                                ;                 bit position (starting at the LSB), and read the same
                                ;                 location, verifying the write.
                                ;
                                ;             3.  Repeat step 2  until the  highest address in the  RAM
                                ;                 is reached.
                                ;
                                ;             4.  Starting again at the lowest address, repeat steps  2
                                ;                 and 3 for every bit of the data word.
                                ;
                                ;             5.  Starting again at the lowest address, repeat steps 2,
                                ;                 3,  and 4, but this time putting zeros back into each
                                ;                 bit.
                                ;
                                ;             6.  Now, starting with the highest address, repeat  steps
                                ;                 2,  3,  4,  and 5, but stepping through the memory in
                                ;                 reverse  order,  from  highest  address  to  lowest
                                ;                 address.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       107107 - RAM buffer error
                                ;
                                ;;******************************************************************************
```

```
000505  033746  000001  130455  RAMTST: MOV     #1,R6           %CALL   INCRTN  ;INITIALIZE R6 VALUE /
                                                                                ; INCREMENT TSTCNT(Q) & RETURN.
000506  133745  000100  120501          MOV     #<1024.*16.>,UBAR %CALL RAMZAP  ;GO INITIALIZE RAM

000507  034440  000000  000000          CLR     R0                              ;INITIALIZE OLD BUFFER VALUE
000510  133750  000100  100526          MOV     #<1024.*16.>,R10  %JMP  4$      ;INITIALIZE R10 VALUE (16K)

000511  016542  000034  010000  1$:     XOR     #FAILUR,R2\N                    ;DOES UBAR (R5) = FAILUR?
000512  033445  017005  150517          MOV\F   UBAR,,BAR       %JZRO   2$      ;JUMP IF IT DOES, ELSE
                                                                                ; IS UBAR (R5) NEGATIVE?
000513  016500  030003  110522          XOR     (BUF),R0\N      %JNEG   3$      ;EXIT LOOP IF DONE /
                                                                                ; IS OLD BUFFER VALUE OK?
000514  033441  013001  040046          MOV\F   R1,,BUF         %JNZRO  ERRB2E  ;ERROR IF NOT, ELSE
                                                                                ; WRITE NEW VALUE TO BUFFER.
000515  033445  007005  000000          MOV     UBAR,,BAR                       ;RESET RAM ADDRESS
000516  016501  000003  010000          XOR     (BUF),R1\N                      ;IS NEW BUFFER VALUE OK?
000517  030145  010006  040046  2$:     ADD\F   R6,UBAR         %JNZRO  ERRB2E  ;ERROR IF NOT, ELSE
                                                                                ; SET NEXT RAM ADDRESS.
000520  033442  000005  000000          MOV     UBAR,R2                         ;SAVE RAM ADDRESS IN R2
000521  135542  000300  110511          BIC     #BIT15!BIT14,R2 %JMP    1$      ;GET RID OF JUNK BITS &
                                                                                ; DO NEXT COMPARE.
000522  033440  000001  000000  3$:     MOV     R1,R0                           ;THE NEW BUFFER VALUE IS NOW THE OLD BUFFER VALUE
000523  176541  010200  010527          XOR\L   #100000,R1      %JNZRO  5$      ;COMPUTE NEXT NEW BUFFER VALUE
000524  132446  000006  010000          NEG     R6                              ;CHANGE SENSE OF R6
                                                                                ; DONE WITH TEST (DID R6 GO FROM -1 TO 1) ?
000525  137750  030300  010455          MOV     #<1024.*16.>-1,R10 %JNNEG INCRTN ;RETURN IF DONE /
                                                                                ; R10 HAS NEW ENDING ADDRESS (16k-1).
000526  033741  000001  010000  4$:     MOV     #1,R1                           ;INITIALIZE NEW BUFFER VALUE
000527  033445  007010  100511  5$:     MOV     R10,UBAR,BAR    %JMP    1$      ;RESET STARTING VALUE
```



```
                        ;;*****************************************************************************
                        ;
                        ;LEV01
                        ;       BCAI REGISTER FILE TEST
                        ;
                        ;               The U-PROC will run this test  in  BI  LOOPBACK  mode.
                        ;
                        ;               This  test  checks the BCAI register file buffers from
                        ;               the 2901 processor side (II side) of the internal  BDA
                        ;               bus.    A walking ONE's and ZERO's data pattern will be
                        ;               written and read across all the BCAI buffers  used  by
                        ;               the  BDA.  The BCAI buffers will be written with data,
                        ;               then read back to verify data integrity.
                        ;
                        ;               In order to read the BCAI command/address buffers, the
                        ;               U-PROC  must  set  BIT00 in the UCR.  BIT00 set allows
                        ;               the BCAI command/address buffers to be read back  with
                        ;               the same IOC that was used to load the buffer.
                        ;
                        ;               The BDA uses 2 octaword data wrap-around buffers and 3
                        ;               command/address buffers on the II side of the BCAI.
                        ;
                        ;       Input:
                        ;               R6 contains value for BREG.
                        ;       Output:
                        ;               none
                        ;
                        ; NOTE: Errors in this routine display the following in the SA register;
                        ;       106341 - CMD DONE error
                        ;       106341 - CMD NOT DONE error
                        ;       106341 - BCAI BUFFER error
                        ;
                        ;;*****************************************************************************

000530  013740  007050  135624  BCATST: MOV     #TEMP3,\N,BAR   %CALL   S.STR6  ;STORE BREG IN TEMP3
000531  015546  116001  056237  1$:     BIC\F   #B.NABO,R6\N,BREG %JSTOP BI.STP ;JUMP IF BI STOPPED, ELSE                [mr1001]
                                                                                ; ABORT BI COMMAND (RESET BIIC)
000532  033446  006006  010000          MOV     R6,,BREG                        ;LOAD BREG
000533  176547  000200  000000          XOR\L   #100000,R7                      ;R7 = DATA
000534  013564  014001  040537          BIS\T   #BIT00,CRI\N,UCRD %JNZRO 2$     ;JUMP IF NEXT PATTERN, ELSE
                                                                                ; ENABLE BCAI TEST MODE
000535  015544  004001  120350          BIC     #BIT00,CRI\N,UCRD %CALL RS.BER  ;DISABLE BCAI TEST MODE /
                                                                                ; GO RESET BI BUS ERROR REGISTER
000536  013440  000000  110252          NOP                             %JMP    TSTRTN  ;TEST UER & RETURN.

000537  033740  000025  131077  2$:     MOV     #21.,R0         %CALL   WT.RAM  ;GET # OF ENTRIES IN RAM BUFFER /
                                                                                ; GO WRITE BCAI DATA IN RAM BUFFER
```

```
                                        ;;Move the data patterns from the RAM buffer to the BCAI GPR buffers

000540  033742  000002  010000          MOV     #2.,R2                                  ;OCTAWORD BUFFER COUNT
000541  013700  005523  000000  3$:     MOV     (BUF),BCAID @WRFST                      ;WRITE FIRST WORD TO BUFFER
000542  033750  000007  110545          MOV     #7.,R10         %JMP    5$              ;NUMBER OF NEXT WRITES /
                                                                                        ; INCREMENT TO NEXT RAM ADDRESS
000543  013700  005723  010000  4$:     MOV     (BUF),BCAID @WRNXT                      ;WRITE NEXT WORD
000544  031450  000010  000000          DEC     R10                                     ;DONE WRITING BCAI BUFFER?
000545  130445  017005  010543  5$:     INC     UBAR,,BAR       %JNZRO  4$              ;JUMP IF NOT DONE /
                                                                                        ; INC TO NEXT RAM ADDRESS
000546  031442  000002  000000          DEC     R2                                      ;DONE BOTH BCAI BUFFERS YET?
000547  013440  010000  110551          NOP                     %JZRO   6$              ;JUMP IF DONE /
000550  013440  000640  100541          NOP             @GOWR   %JMP    3$              ;FLIP TO OTHER BCAI OCTAWORD BUFFER
                                                                                        ; BY USING GOWR IOC

                                        ;;Move the data patterns from the RAM buffer to the BCAI CMD/ADR buffers

000551  013440  110000  006237  6$:     NOP                             %JSTOP BI.STP   ;JUMP IF BI STOPPED                             [mr1001]
000552  015546  166001  000043          BIC     #B.NABO,R6\N,BREG       %JCDONE ERR81B  ;ERROR IF CMD DONE / ABORT BI COMMAND
000553  033446  006006  130275          MOV     R6,,BREG                %CALL WASTE     ;RELOAD BREG /
                                                                                        ; WASTE SOME TIME FOR CMD DONE
000554  015544  164001  110043          BIC     #BIT00,CRI\N,UCRD       %JNCDONE ERR81B ;ERROR IF CMD IS NOT DONE /
                                                                                        ; DISABLE BCAI TEST MODE
000555  013700  005563  010000          MOV     (BUF),BCAID @LADD                       ;WRITE BCAI LO ADDR BUFFER
000556  130445  007005  010000          INC     UBAR,,BAR                               ;INCREMENT TO NEXT RAM ADDRESS
000557  013700  005763  000000          MOV     (BUF),BCAID @HADD                       ;WRITE BCAI HI ADDR BUFFER
000560  130445  007005  010000          INC     UBAR,,BAR                               ;INCREMENT TO NEXT RAM ADDRESS
000561  033702  000003  010000          MOV     (BUF),R2                                ;GET COMMAND BUFFER PATTERN FROM RAM
000562  034542  000017  125621          AND     #17,R2          %CALL   S.STR2  ;SAVE BITS 3 - 0 /
                                                                                        ; STORE DATA PATTERN
000563  033445  007005  000000          MOV     UBAR,,BAR                               ;RELOAD RAM ADDRESS
000564  013700  005543  000000          MOV     (BUF),BCAID @LCOM                       ;WRITE BCAI COMMAND BUFFER
000565  130445  007005  010000          INC     UBAR,,BAR                               ;INCREMENT TO NEXT RAM ADDRESS
000566  033702  000003  010000          MOV     (BUF),R2                                ;GET LO NEXT PAGE BUFFER PATTERN FROM RAM
000567  134542  000376  125621          AND     #177000,R2      %CALL   S.STR2  ;SAVE BITS 15 - 9
                                                                                        ; STORE DATA PATTERN
000570  033445  007005  000000          MOV     UBAR,,BAR                               ;RELOAD RAM ADDRESS
000571  013700  005443  010000          MOV     (BUF),BCAID @LNPAG                      ;WRITE BCAI NEXT PAGE BUFFER
000572  130445  007005  010000          INC     UBAR,,BAR                               ;INCREMENT TO NEXT RAM ADDRESS
000573  013700  005763  000000          MOV     (BUF),BCAID @HADD                       ;WRITE BCAI HI ADDR IN NEXT PAGE BUFFER
000574  013544  004001  131104          BIS     #BIT00,CRI\N,UCRD %CALL S.BUFR  ;ENABLE BCAI TEST MODE /
                                                                                        ; SETUP BUFFERS TO READ BCAI FILE

                                        ;;READ data from BCAI GPR buffers to a RAM buffer

000575  033742  000002  010000          MOV     #2.,R2                                  ;OCTAWORD BUFFER COUNT
000576  030145  007000  000000          ADD     R0,UBAR,BAR                             ;POINT TO RECIEVED DATA BUFFER IN RAM
000577  033750  000007  010000  7$:     MOV     #7.,R10                                 ;NUMBER OF NEXT READS
000600  013440  000500  000000          NOP                             @RDFST          ;READ FIRST WORD FROM BUFFER
                                                                                        ; FLIP TO OTHER BCAI OCTAWORD BUFFER
000601  013700  003715  010000  8$:     MOV     (BCAIS),\N,BUF  @RDNXT                  ;GET DATA FROM BUFFER /
                                                                                        ; READ NEXT WORD.
000602  031450  000010  000000          DEC     R10                                     ;DONE READING BCAI BUFFER?
```



```
000603  130445  017005  010601          INC     UBAR,,BAR       %JNZRO  8$      ;JUMP IF NOT DONE
                                                                                ; INC TO NEXT RAM ADDRESS
000604  013700  003015  000000          MOV     (BCAIS),\N,BUF                  ;GET DATA FROM BCAI BUFFER
000605  031442  000002  000000          DEC     R2                              ;DONE BOTH BCAI BUFFERS YET?
000606  130445  017005  000577          INC     UBAR,,BAR       %JNZRO  7$      ;JUMP IF NOT DONE /
                                                                                ; INC TO NEXT RAM ADDRESS
```

```
                                        ;;READ data from BCAI CMD/ADR buffers to a RAM buffer

000607  013440  000560  000000          NOP                 @LADD           ;READ BCAI LO ADDR BUFFER
000610  013700  003775  010000          MOV     (BCAIS),\N,BUF @HADD        ;GET DATA FROM BCAI BUFFER /
                                                                            ; READ BCAI HI ADDR BUFFER
000611  130445  007005  010000          INC     UBAR,,BAR                   ;INCREMENT TO NEXT RAM ADDRESS
000612  013700  003555  010000          MOV     (BCAIS),\N,BUF @LCOM        ;GET DATA FROM BCAI BUFFER /
                                                                            ; READ BCAI COMMAND BUFFER
000613  130445  007005  010000          INC     UBAR,,BAR                   ;INCREMENT TO NEXT RAM ADDRESS
000614  013700  003455  000000          MOV     (BCAIS),\N,BUF @LNPAG       ;GET DATA FROM BCAI BUFFER /
                                                                            ; READ BCAI NEXT PAGE BUFFER
000615  130445  007005  010000          INC     UBAR,,BAR                   ;INCREMENT TO NEXT RAM ADDRESS
000616  013700  003775  010000          MOV     (BCAIS),\N,BUF @HADD        ;GET DATA FROM BCAI BUFFER /
                                                                            ; READ BCAI HI ADDR BUFFER
000617  130445  007005  010000          INC     UBAR,,BAR                   ;INCREMENT TO NEXT RAM ADDRESS
000620  013700  003015  131104          MOV     (BCAIS),\N,BUF  %CALL   S.BUFR  ;GET DATA FROM BCAI BUFFER /
                                                                            ; SETUP BUFFERS TO COMPARE DATA

                                        ;;Compare the EXPECTED & RECIEVED data of the BCAI buffers

000621  033750  000025  010000          MOV     #21.,R10                    ;NUMBER OF BUFFERS TO COMPARE
000622  033702  010003  140627  9$:     MOV\F   (BUF),R2        %JZRO   10$ ;JUMP IF DONE /
                                                                            ; SAVE EXPECTED DATA
000623  010145  007000  010000          ADD     R0,UBAR\N,BAR               ;ADD RING LENGTH TO BAR &
                                                                            ; POINT TO RECIEVED DATA BUFFER IN RAM
000624  036502  000003  000000          XOR     (BUF),R2                    ;EXPECTED DATA SAME AS RECIEVED DATA?
000625  130445  017005  010043          INC     UBAR,,BAR       %JNZRO  ERRB1B  ;ERROR IF NOT THE SAME /
                                                                            ; INC TO NEXT RAM ADDRESS
000626  031450  000010  110622          DEC     R10             %JMP    9$  ;DECREMENT RING LENGTH.

000627  013440  000700  110531  10$:    NOP             @RDNXT  %JMP    1$  ;DO IT AGAIN /
                                                                            ; FLIP BACK TO ORIGINAL BCAI BUFFER
                                                                            ; BACK BY USING RDNXT IOC
```



```
                        ;;*****************************************************************************
                        ;
                        ;LEVO1
                        ;       BI PARITY TEST
                        ;
                        ;               The U-PROC will run this test  in  BI  LOOPBACK  mode.
                        ;
                        ;       Force WRITE PARITY ERROR 2 (RAM to BCAI buffer write).
                        ;
                        ;               The first test will verify that  parity  detection  is
                        ;               correct  from  BDA RAM to the BCAI.  Data being xfered
                        ;               from RAM to the BCAI has parity  associated  with  it.
                        ;               When read from the BCAI (during the write to the BIIC)
                        ;               the parity is checked for correctness.  To  force  the
                        ;               parity  vector, Data with bad parity is written from a
                        ;               2901 GPR (which does not have parity  associated  with
                        ;               it)  to  the  BCAI.   The  data  is read from the BCAI
                        ;               register (during the write to a BIIC GPR).  This  will
                        ;               cause  the U-PROC and D-PROC to vector to high memory.
                        ;               An error will  result  in  this  test  if  the  U-PROC
                        ;               vectors to an incorrect address in ROM or if no vector
                        ;               occurs.
                        ;
                        ;               NOTE:   The data will be passed from the BCAI  to  the
                        ;                       BIIC,  regardless  of  the parity error.  This
                        ;                       will be verified in the BIIC  to  BCAI  parity
                        ;                       test.
                        ;
                        ;       Force BCI PARITY ERROR 1 (BCAI to RAM buffer write).
                        ;
                        ;               The second test will verify that parity  detection  is
                        ;               correct  from the BCAI to BDA RAM.  As data is written
                        ;               from a BCAI register to RAM, parity is generated  from
                        ;               the RAM interface and is compared with the parity from
                        ;               the BCAI.  If not correct, an error  will  occur.   In
                        ;               this test, an all 1's data word is written to the NEXT
                        ;               PAGE ADDRESS buffer in the BCAI  by  using  the  LNPAG
                        ;               IOC.   The  U-PROC  sets BIT00 in the UCR to allow the
                        ;               BCAI command/address buffers to be read back with  the
                        ;               same  IOC  that loads them.  The data is read from the
                        ;               NEXT PAGE ADDRESS buffer in the the BCAI by using  the
                        ;               LNPAG  IOC.   This  will cause the U-PROC to vector to
                        ;               high memory.  An error will result in this test if the
                        ;               U-PROC vectors to an incorrect address in ROM or if no
                        ;               vector occurs.
```

```
                                ;       Force READ PARITY ERROR 2 (BIIC READ DATA cycle).
                                ;
                                ;               The third test will verify that parity detection is
                                ;               correct from the BIIC (read data) to the BCAI.  The
                                ;               BCAI generates byte parity based on data coming from
                                ;               the BIIC and compares it to the parity line from the
                                ;               BIIC.  If they do not agree, an error results.  To
                                ;               cause the parity error, the U-PROC sets BIT01 in the
                                ;               UCR to enable BI parity test.  The data with bad
                                ;               parity is read from a BIIC GPR.  This will cause the
                                ;               U-PROC and D-PROC to vector to high memory.  An error
                                ;               will result in this test if the U-PROC vectors to an
                                ;               incorrect address in ROM or if no vector occurs.
                                ;
                                ;               NOTE:   The BIIC GPR data with parity was previously
                                ;                       written during the RAM to BCAI PARITY TEST.
                                ;
                                ;       Force PARITY ERRORS (BIIC COMMAND/ADDRESS cycle).                       BEG [mr1002]
                                ;
                                ;               The fourth test is executed in MANUFACTURING ONLY.  It
                                ;               will verify that parity detection occurs during a BIIC
                                ;               COMMAND/ADDRESS cycle.  An error will result if the
                                ;               CPE bit (Command Parity Error) is not set in the BI
                                ;               BER (Bus Error Reg).
                                ;
                                ;       Force PARITY ERRORS (BIIC WRITE DATA cycle).
                                ;
                                ;               The fifth test is executed in MANUFACTURING ONLY.  It
                                ;               will verify that parity detection occurs during a BIIC
                                ;               WRITE DATA cycle.  An error will result if the SPE
                                ;               bit (Slave Parity Error) is not set in the BI BER (Bus
                                ;               Error Reg).                                             END [mr1002]
                                ;
                                ;       Input:
                                ;               R6 contains value for BREG.
                                ;       Output:
                                ;               none
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;       106342 - BI PARITY error (LOOPBACK MODE)
                                ;
                                ;;*************************************************************************

000630  033740  000370  120353  BIPE:   MOV     #BIGPR2,R0      %CALL   CHKPAR  ;GET OFFSET TO BIIC GPR 2 /
                                                                                ; GO CHECK PARITY
000631  030540  040004  040632          ADD\F   #4,R0           %TLSB           ;CONTINUE IF ODD # OF BITS ARE SET,
                                                                                ; FOR THE BI COMMAND/ADDRESS CYCLE, ELSE
                                                                                ; INC TO NEXT GPR ADRS TO MAKE BITS ODD,
                                                                                ; ALLOW CORRECT COMMAND/ADDRESS TO BIIC
000632  013440  000000  130265          NOP                     %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS.

                                        ;;Test WRITE PARITY ERROR 2 (U-PROC vectors to 7772, D-PROC vectors to 7762)
```



```
000633  033740  000004  000000          MOV     #BIWRCM,R0                      ;DO WRITE COMMAND
000634  033440  005540  000000          MOV     R0,,BCAID   @LCOM               ;LOAD COMMAND
000635  033440  005520  000000          MOV     R0,,BCAID   @WRFST              ;WRITE FIRST BUFFER
000636  037140  005740  130640          XNOR    R0,R0,BCAID @LBWR %CALL 1$      ;INITIALIZE WAIT LOOP COUNT /
                                                                                ; WRITE LAST BUFFER AND DO THE WRITE /
                                                                                ; SETUP VECTOR RETURN ADDRESS
000637  036547  000003  100642          XOR     #ER.WP2,R7      %JMP    2$      ;TEST RESULTS OF R7 DURING BI PARITY VECTOR
000640  031440  010000  000640  1$:     DEC     R0              %JNZR0  1$      ;WAIT FOR PARITY ERROR VECTOR
000641  013440  000000  100043          NOP                     %JMP    ERRBIB  ;ERROR IF NOT FORCED TO VECTOR TABLE
000642  033446  016006  010043  2$:     MOV     R6,,BREG        %JNZR0  ERRBIB  ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES / RELOAD BREG
```

```
                                        ;;Test BCI PARITY ERROR 1 (U-PROC vectors to 7773)

000643  013740  007050  010000          MOV     #TEMP3,\N,BAR                   ;POINT TO TEMP3 ADDR IN RAM AS TEMP BUFFER
000644  037140  005440  000000          XNOR    RO,RO,BCAID @LNPAG              ;INITIALIZE WAIT LOOP COUNT /
                                                                                ; WRITE BCAI LO NEXT PAGE BUFFER
000645  013544  004001  000000          BIS     #BITOO,CRI\N,UCRD               ;ENABLE READ OF BCAI CMD & ADDR REGISTERS
000646  013440  000440  000000          NOP                 @LNPAG              ;READ BCAI LO NEXT PAGE BUFFER
000647  013700  003015  130651          MOV     (BCAIS),\N,BUF  %CALL   3$      ;GET DATA BCAI LO NEXT PAGE BUFFER /
                                                                                ; SETUP VECTOR RETURN ADDRESS
000650  036547  000003  100653          XOR     #ER.BP1,R7      %JMP    4$      ;TEST RESULTS OF R7 DURING BCI PARITY VECTOR
000651  031440  010000  000651  3$:     DEC     RO              %JNZRO  3$      ;WAIT FOR PARITY ERROR VECTOR
000652  013440  000000  100043          NOP                     %JMP    ERRB1B  ;ERROR IF NOT FORCED TO VECTOR TABLE
000653  015544  014001  000043  4$:     BIC     #BITOO,CRI\N,UCRD %JNZRO ERRB1B ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES /
                                                                                ; DISABLE READ OF BCAI CMD & ADDR REGISTERS

                                        ;;Test READ PARITY ERROR 2 (U-PROC vectors to 7771, D-PROC vectors to 7761)

000654  033740  000370  120353          MOV     #BIGPR2,RO      %CALL   CHKPAR  ;GET OFFSET TO BIIC GPR 2 /
                                                                                ; GO CHECK PARITY
000655  030540  040004  050656          ADD\F   #4,RO           %TLSB           ;CONTINUE IF ODD # OF BITS ARE SET,
                                                                                ; FOR THE BI COMMAND/ADDRESS CYCLE, ELSE
                                                                                ; INC TO NEXT GPR ADRS TO MAKE BITS ODD,
                                                                                ; ALLOW CORRECT COMMAND/ADDRESS TO BIIC
000656  013544  004002  130265          BIS     #BITO1,CRI\N,UCRD %CALL BRGADR  ;ENABLE BI PARITY TEST /
                                                                                ; GO LOAD BIIC REGISTER ADDRESS.
000657  033740  000001  000000          MOV     #BIRDCM,RO                      ;GO DO READ COMMAND
000660  033440  005540  000000          MOV     RO,,BCAID @LCOM                 ;LOAD COMMAND
000661  037140  000420  120663          XNOR    RO,RO       @GORD %CALL 5$      ;INITIALIZE WAIT LOOP COUNT /
                                                                                ; DO THE READ /
                                                                                ; SETUP VECTOR RETURN ADDRESS
000662  036547  000003  100665          XOR     #ER.RP2,R7      %JMP    6$      ;TEST RESULTS OF R7 DURING BI PARITY VECTOR
000663  031440  010000  010663  5$:     DEC     RO              %JNZRO  5$      ;WAIT FOR PARITY ERROR VECTOR
000664  013440  000000  100043          NOP                     %JMP    ERRB1B  ;ERROR IF NOT FORCED TO VECTOR TABLE
000665  015544  014002  000043  6$:     BIC     #BITO1,CRI\N,UCRD %JNZRO ERRB1B ;ERROR IF INCORRECT RESULTS DURING
                                                                                ; VECTOR ROUTINES /
                                                                                ; DISABLE BI PARITY TEST.
000666  033446  006006  130350          MOV     R6,,BREG        %CALL   RS.BER  ;RELOAD BREG /
                                                                                ; GO RESET BI BUS ERROR REGISTER
```



```
                                        ;;** THIS IS A MANUFACTURING TEST ONLY **                                        [mr1002]

                                        ;;Force PARITY ERROR during BI COMMAND/ADDRESS cycle

000667  013440  050000  110671          NOP                     %JNTEST 7$      ;CHECK EXT.TEST_L FOR TOGGLE MODE
000670  013440  050000  000252          NOP                     %JTEST  TSTRTN  ;RETURN IF EXT.TEST_L DID NOT TOGGLE
000671  013440  050000  110252  7$:     NOP                     %JNTEST TSTRTN  ;RETURN IF EXT.TEST_L DID NOT TOGGLE
000672  033740  000370  120353          MOV     #BIGPR2,RO      %CALL   CHKPAR  ;GET OFFSET TO BIIC GPR 2 /
                                                                                ; GO CHECK PARITY
000673  030540  040004  140674          ADD\F   #4,RO           %TNLSB          ;CONTINUE IF EVEN # OF BITS ARE SET,
                                                                                ; FOR THE BI COMMAND/ADDRESS CYCLE, ELSE
                                                                                ; INC TO NEXT GPR ADRS TO MAKE BITS EVEN.
                                                                                ; ALLOW BAD ADDRESS TO BIIC
000674  013544  004002  130265          BIS     #BITO1,CRI\N,UCRD %CALL BRGADR  ;ENABLE BI PARITY TEST /
                                                                                ; GO LOAD BIIC REGISTER ADDRESS.
000675  033740  000001  000000          MOV     #BIRDCM,RO                      ;DO READ COMMAND
000676  033440  005540  000000          MOV     RO,,BCAID @LCOM                 ;LOAD COMMAND
000677  013440  000420  120275          NOP                 @GORD %CALL WASTE   ;DO THE READ /
                                                                                ; GO WASTE SOME TIME
000700  037140  000000  000000          XNOR    RO,RO                           ;SETUP GROSS TIMEOUT LOOP (91ms)
000701  031440  160000  010704  8$:     DEC     RO              %JCDONE 9$      ;WAIT UNTIL COMMAND IS DONE
000702  013440  010000  010701          NOP                     %JNZRO  8$      ;DECREMENT TIMEOUT LOOP
000703  013440  000000  110042          NOP                     %JMP    ERRB1T  ;ERROR IF COMMAND TIMEOUT

000704  013440  000500  000000  9$:     NOP                 @RDFST              ;READ 1ST BUFFER
000705  013740  007271  000000          MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
000706  033701  003715  010000          MOV     (BCAIS),R1,BUF @RDNXT           ;GET DATA FROM 1ST BCAI BUFFER /
                                                                                ; READ NEXT BUFFER.
000707  013740  007272  000000          MOV     #RGDATH,\N,BAR                  ;POINT TO HI REG DATA LOC IN RAM
000710  033702  003015  000000          MOV     (BCAIS),R2,BUF                  ;GET DATA FROM NEXT BCAI BUFFER
000711  015544  144002  100043          BIC     #BITO1,CRI\N,UCRD %JNMERR ERRB1B ;ERROR IF NO ERROR DETECTED /
                                                                                ; DISABLE BI PARITY TEST.
000712  015546  006001  000000          BIC     #B.NABO,R6\N,BREG               ;RESET MASTER ERROR
000713  033446  006006  010000          MOV     R6,,BREG                        ;LOAD BREG
000714  033740  000010  130313          MOV     #BIBER,RO       %CALL   RDCMD   ;READ BI BUS ERROR REGISTER
000715  014542  000200  000000          BIT     #<B.CPE>,R2                     ;CHECK FOR COMMAND PARITY ERROR
000716  013440  010000  110043          NOP                     %JZRO   ERRB1B  ;ERROR IF CPE NOT DETECTED
000717  013440  000000  120350          NOP                     %CALL   RS.BER  ;GO RESET BI BUS ERROR REGISTER
```

```
                                        ;;** THIS IS A MANUFACTURING TEST ONLY **                                    [mr1002]

                                        ;;Force PARITY ERROR during BI WRITE DATA

000720  033740  000370  120353        MOV     #BIGPR2,R0      %CALL   CHKPAR  ;GET OFFSET TO BIIC GPR 2 /
000721  030540  040004  040722        ADD\F   #4,R0           %TLSB           ; GO CHECK PARITY
                                                                              ;CONTINUE IF ODD # OF BITS ARE SET,
                                                                              ; FOR THE BI COMMAND/ADDRESS CYCLE, ELSE
                                                                              ; INC TO NEXT GPR ADRS TO MAKE BITS ODD,
                                                                              ; ALLOW CORRECT COMMAND/ADDRESS TO BIIC
000722  013544  004002  000000        BIS     #BIT01,CRI\N,UCRD               ;ENABLE BI PARITY TEST
000723  034441  000001  130265        CLR     R1              %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS /
                                                                              ; INITIALIZE DATA PATTERNS
000724  133742  000300  010000        MOV     #BIT15+BIT14,R2                 ; TO ALLOW BAD PARITY DURING BI WRITE DATA.
000725  033740  000004  000000        MOV     #BIWRCM,R0                      ;DO WRITE COMMAND
000726  033440  005540  000000        MOV     R0,,BCAID       @LCOM           ;LOAD COMMAND
000727  013740  007271  000000        MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
000730  033441  003001  135621        MOV     R1,,BUF         %CALL   S.STR2  ;STORE REGISTER DATA
000731  013740  007271  000000        MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
000732  013700  005523  000000        MOV     (BUF),BCAID     @WRFST          ;WRITE FIRST BUFFER
000733  013740  007272  000000        MOV     #RGDATH,\N,BAR                  ;POINT TO HI REG DATA LOC IN RAM
000734  013700  005743  130275        MOV     (BUF),BCAID     @LBWR %CALL WASTE ;WRITE LAST BUFFER AND DO THE WRITE /
000735  037140  000000  000000        XNOR    R0,R0                           ;SETUP GROSS TIMEOUT LOOP (91mS)
000736  031440  160000  000741  10$:  DEC     R0                      %JCDONE 11$     ;JUMP WHEN COMMAND DONE
000737  013440  010000  000736        NOP                             %JNZRO  10$     ;DECREMENT TIMEOUT LOOP
000740  013440  000000  110042        NOP                             %JMP    ERRB1T  ;ERROR IF COMMAND TIMEOUT

000741  015544  144002  100043  11$:  BIC     #BIT01,CRI\N,UCRD %JNMERR ERRB1B ;ERROR IF NO ERROR DETECTED /
                                                                              ; DISABLE BI PARITY TEST.
000742  015546  006001  000000        BIC     #B.NABO,R6\N,BREG               ;RESET MASTER ERROR
000743  033446  006006  010000        MOV     R6,,BREG                        ;RELOAD BREG
000744  033740  000010  130313        MOV     #BIBER,R0       %CALL   RDCMD   ;READ BI BUS ERROR REGISTER
000745  014542  000100  000000        BIT     #<B.SPE>,R2                     ;CHECK FOR SLAVE PARITY ERROR
000746  013440  010000  110043        NOP                             %JZRO   ERRB1B  ;ERROR IF SPE NOT DETECTED
000747  013440  000000  120350        NOP                             %CALL   RS.BER  ;GO RESET BI BUS ERROR REGISTER
000750  013440  000000  110252        NOP                             %JMP    TSTRTN  ;TEST UER & RETURN.
```



```
                                ;;******************************************************************************
                                ;
                                ;LEV01
                                ;
                                ;       BIIC BUFFER TEST
                                ;
                                ;               The U-PROC will run the BIIC Buffer test  in  both  BI
                                ;               LOOPBACK  and  NORMAL modes.  This test will write and
                                ;               read a walking ONE's and ZERO's data pattern across  a
                                ;               BIIC  GPR  (General Purpose Register 2).  The register
                                ;               will be written with data, then read  back  to  verify
                                ;               data  integrity.    All BIIC register xfers are done in
                                ;               LONGWORD mode.
                                ;
                                ;       Input:
                                ;               R6 contains value for BREG.
                                ;       Output:
                                ;               none
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106343 - BIIC BUFFER error (LOOPBACK MODE)
                                ;       106344 - BIIC BUFFER error (NORMAL MODE)
                                ;
                                ;;******************************************************************************

000751  013740  007050  135624  BIITST: MOV     #TEMP3,\N,BAR   %CALL   S.STR6  ;STORE BREG IN TEMP3
000752  176547  100200  070252  1$:     XOR\LF  #100000,R7      %CFTEST TSTRTN  ;TEST UER IF FAST SELF-TEST ENABLED, ELSE
                                                                              ; LOAD STARTING DATA
000753  013440  010000  000756          NOP                             %JNZRO 2$       ;JUMP IF NOT DONE YET                   [mr1001]
000754  015546  116001  056237          BIC\F   #B.NABO,R6\N,BREG %JSTOP BI.STP ;JUMP IF BI STOPPED, ELSE                       [mr1001]
                                                                              ; ABORT BI CMD
000755  033446  006006  100252          MOV     R6,,BREG        %JMP    TSTRTN  ;LOAD BREG / TEST UER & RETURN

000756  033740  000002  131104  2$:     MOV     #2.,R0          %CALL   S.BUFR  ;GET # OF PATTERNS TO GENERATE /
                                                                              ; SETUP BUFFERS
000757  033442  000007  010000          MOV     R7,R2                           ;MOVE R7 DATA TO R2
000760  033442  013002  140764  3$:     MOV\F   R2,,BUF         %JZRO   4$      ;MOVE DATA TO RAM BUFFER IF NOT DONE
000761  176542  000200  000000          XOR\L   #100000,R2                      ;CHANGE VALUE IN NEXT ADDRESS
000762  130445  007005  010000          INC     UBAR,,BAR                       ;INCREMENT TO NEXT RAM ADDRESS
000763  031450  000010  100760          DEC     R10             %JMP    3$      ;ALL DONE? LOOP

                                        ;;Move the data patterns from the RAM buffer to the BCAI buffers

000764  013440  000000  131104  4$:     NOP                             %CALL   S.BUFR  ;SETUP BUFFERS
000765  033740  000370  130265          MOV     #BIGPR2,R0      %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS
000766  033740  000004  000000          MOV     #BIWRCM,R0                      ;GET WRITE COMMAND
000767  033440  005540  000000          MOV     R0,,BCAID       @LCOM           ;LOAD COMMAND
000770  033445  007005  000000          MOV     UBAR,,BAR                       ;POINT TO EXPECTED DATA BUFFER IN RAM
000771  013700  005523  000000          MOV     (BUF),BCAID @WRFST              ;WRITE FIRST BUFFER
000772  130445  117005  016237          INC     UBAR,,BAR       %JSTOP  BI.STP   ;JUMP IF BI STOPPED /                  [mr1001]
                                                                              ; INC TO NEXT RAM ADDRESS
000773  013700  005743  120323          MOV     (BUF),BCAID @LBWR %CALL WTCMDD  ;WRITE LAST BUFFER AND DO THE WRITE /
                                                                              ; GO WAIT FOR CMD COMPLETE & ERROR FREE
```

```
                                        ;;READ data from BCAI GPR buffers to a RAM buffer

000774  013440  000000  131104            NOP                     %CALL   S.BUFR  ;SETUP BUFFERS
000775  030545  000002  010000            ADD     #2.,UBAR                        ;POINT TO RECIEVED DATA BUFFER
000776  033740  000370  130265            MOV     #BIGPR2,R0      %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS
000777  033740  000001  000000            MOV     #BIRDCM,R0                      ;GET READ COMMAND
001000  033440  115540  006237            MOV     R0,,BCAID @LCOM %JSTOP  BI.STP  ;JUMP IF BI STOPPED / LOAD COMMAND      [mr1001]
001001  013440  000420  130323            NOP             @GORD %CALL   WTCMDD    ;DO THE READ /
                                                                                ; GO WAIT FOR CMD COMPLETE & ERROR FREE
001002  033445  007505  000000            MOV     UBAR,,BAR @RDFST                ;READ FIRST BCAI BUFFER /
                                                                                ; POINT TO RECIEVED DATA BUFFER IN RAM
001003  013700  003715  010000            MOV     (BCAIS),\N,BUF @RDNXT           ;GET DATA FROM BCAI BUFFER /
                                                                                ; READ NEXT BUFFER.
001004  130445  007005  010000            INC     UBAR,,BAR                       ;INC TO NEXT RAM ADDRESS
001005  013700  003015  000000            MOV     (BCAIS),\N,BUF                  ;GET DATA FROM BCAI BUFFER

                                        ;;Compare the EXPECTED & RECIEVED data of the GPR buffers

001006  033740  000002  131104            MOV     #2.,R0          %CALL   S.BUFR  ;SETUP BUFFERS TO COMPARE DATA
001007  033702  010003  110752  5$:       MOV     (BUF),R2        %JZR0   1$      ;JUMP IF DONE /
                                                                                ; SAVE EXPECTED DATA
001010  010145  007000  010000            ADD     R0,UBAR\N,BAR                   ;ADD RING LENGTH TO BAR &
                                                                                ; POINT TO RECIEVED DATA BUFFER IN RAM
001011  036502  000003  000000            XOR     (BUF),R2                        ;EXPECTED DATA SAME AS RECIEVED DATA?
001012  130445  017005  010043            INC     UBAR,,BAR       %JNZR0  ERRB1B  ;ERROR IF NOT THE SAME /
                                                                                ; INC TO NEXT RAM ADDRESS
001013  031450  000010  111007            DEC     R10             %JMP    5$      ;DECREMENT RING LENGTH.
```



```
                                        ;;*****************************************************************************
                                        ;
                                        ;LEV01
                                        ;
                                        ;       POLL TEST (NORMAL MODE)
                                        ;
                                        ;               This test issues a  read to the IP register (BI GPR 0)
                                        ;               and checks to see that the POLL test condition is set.
                                        ;               An error results if the POLL test  condition  does not
                                        ;               set.  The test  then  issues  a BI abort  command  and
                                        ;               checks to see that  the POLL test condition is  reset.
                                        ;               An error results if the POLL test  condition does  not
                                        ;               reset.
                                        ;
                                        ;       Input:
                                        ;               R6 contains value for BREG.
                                        ;       Output:
                                        ;               none
                                        ;
                                        ; NOTE: Errors in this routine display the following in the SA register;
                                        ;       106345 - POLL TEST error (NORMAL MODE)
                                        ;
                                        ;;*****************************************************************************

001014  033740  000050  120313  POLTST: MOV     #BCICSR,R0      %CALL   RDCMD   ;READ BCI CONTROL REGISTER
001015  133541  000040  010000          BIS     #STOPEN,R1                      ;SET STOPEN BIT &
001016  033541  000210  010000          BIS     #<BICSREN+RTOEVEN>,R1           ;BICSREN+RTOEVEN BIT IN BCICSR          [mjt08]
                                                                                ; TO ENABLE NODE SELECT FOR STOP CMD.
001017  033740  000050  120302          MOV     #BCICSR,R0      %CALL   WRCMD   ;WRITE BCI CONTROL REGISTER
001020  033740  000362  130313          MOV     #IPREG,R0       %CALL   RDCMD   ;READ IP REGISTER (BI GPR 0)
                                                                                ; TO SET POLL
001021  013440  150000  110043          NOP                     %JNPOLL ERRB1B  ;ERROR IF POLL WASN'T SET, ELSE
001022  015546  116001  056237          BIC\F   #B.NABO,R6\N,BREG %JSTOP BI.STP ;JUMP IF BI STOPPED, ELSE               [mr1001]
                                                                                ; ABORT BI CMD
001023  033446  006006  130275          MOV     R6,,BREG        %CALL WASTE     ;RELOAD BREG /
                                                                                ; WASTE SOME TIME TO ALLOW
                                                                                ; RESET OF POLL SIGNAL.
001024  013440  150000  000043          NOP                     %JPOLL  ERRB1B  ;ERROR IF POLL WASN'T RESET
001025  013440  000000  110252          NOP                     %JMP    TSTRTN  ;TEST UER & RETURN
```

```
                                ;;*****************************************************************************
                                ;
                                ;LEV01
                                ;
                                ;       BI MEMORY TEST (NORMAL MODE)
                                ;
                                ;               This test is executed in MANUFACTURING ONLY.  The test          [mr1002]
                                ;               writes  and  reads 3 OCTAWORDS  in BI MEMORY  (must be
                                ;               NODE 15.).  All 3 OCTAWORDS  will  be written  with  a
                                ;               walking  ONE's and ZERO's data pattern,  then  read to
                                ;               verify data integrity.
                                ;
                                ;       Input:
                                ;               none
                                ;       Output:
                                ;               none
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106347 - BI MEMORY error (NORMAL MODE)
                                ;
                                ;;*****************************************************************************

                                        ;;** THIS IS A MANUFACTURING TEST ONLY **                                [mr1002]

001026  013440  050000  101030  BIMEM:  NOP                     %JNTEST 1$      ;CHECK EXT.TEST_L FOR TOGGLE MODE
001027  013440  050000  000252          NOP                     %JTEST  TSTRTN  ;RETURN IF EXT.TEST_L DID NOT TOGGLE
001030  013440  050000  110252  1$:     NOP                     %JNTEST TSTRTN  ;RETURN IF EXT.TEST_L DID NOT TOGGLE
001031  033741  000017  120254          MOV     #15.,R1         %CALL   SHFID   ;ASSUME NODE ID 15. FOR BI MEMORY
                                                                                ; GO SHIFT ID INTO CORRECT POSITION

                                        ;;Read BICSR in BI MEMORY to check STS bit status

001032  033740  000004  130313          MOV     #BICSR,R0       %CALL   RDCMD   ;READ BI CONTROL REGISTER
001033  114541  000010  010000          BIT     #<STS>,R1                       ;WAS SELF-TEST SUCCESSFUL?
001034  013440  010000  110043          NOP                     %JZRO   ERRB18  ;ERROR IF NOT

                                        ;;Setup STARTING and ENDING BI MEMORY address registers

001035  033742  000000  000000          MOV     #0,R2                           ;LOAD HI STARTING ADDRESS
001036  033741  000000  000000          MOV     #0,R1                           ;LOAD LO STARTING ADDRESS
001037  033740  000040  130302          MOV     #BISADR,R0      %CALL   WRCMD   ;WRITE BI STARTING ADDRESS REG
001040  033742  000010  010000          MOV     #10,R2                          ;LOAD HI ENDING ADDRESS
001041  033741  000000  000000          MOV     #0,R1                           ;LOAD LO ENDING ADDRESS
001042  033740  000044  120302          MOV     #BIEADR,R0      %CALL   WRCMD   ;WRITE BI ENDING ADDRESS REG

                                        ;;Setup OCTAWORD count, starting BI MEMORY address & starting DATA pattern

001043  033740  000030  010000  2$:     MOV     #<8.*3>,R0                      ;SETUP FAKE RING LENGTH (3 Octawords)
001044  073746  000340  010000          MOV\L   #<700/2>,IUAR                   ;SETUP STARTING BI MEMORY ADDRESS (700)
001045  073752  000360  000000          MOV\L   #<740/2>,R12                    ;SETUP LAST STARTING BI MEMORY ADDRESS (740)
001046  176547  000200  000000  3$:     XOR\L   #100000,R7                      ;R7 = DATA
001047  016152  010006  041051          XOR\F   IUAR,R12\N      %JNZRO  4$      ;JUMP IF DATA NON-ZERO, ELSE
                                                                                ; DID WE DO LAST STARTING BI ADDRESS?
001050  030546  010002  141070          ADD\F   #2,IUAR         %JZRO   6$      ;EXIT IF YES, ELSE
```



```
                                                                                ; NEXT STARTING BI MEMORY ADDRESS
```

```
                                        ;;Setup to WRITE from a RAM buffer to HOST memory

001051  013740  007050  135624  4$:   MOV     #TEMP3,\N,BAR   %CALL   S.STIU  ;POINT TO TEMP3 ADDR IN RAM /
                                                                               ; SAVE BI MEMORY ADDRESS
001052  013740  007140  010000        MOV     #UBURST,\N,BAR                   ;POINT UBURST IN RAM
001053  013740  003002  121077        MOV     #2,\N,BUF       %CALL   WT.RAM  ;GO WRITE DATA IN RAM BUFFER
001054  033741  000105  136245        MOV     #EDSEED,R1      %CALL   WT.DIAG ;SETUP EDC SEED AND
                                                                               ; GO WRITE DATA TO HOST MEMORY.
001055  033440  010010  000043        MOV     R10,R0          %JNZRO  ERRB1B  ;ERROR IF NOT 0 /
                                                                               ; RESTORE RING LENGTH TO R0

                                        ;;Setup to READ from HOST memory to a RAM buffer

001056  013440  000000  131104        NOP                     %CALL   S.BUFR  ;SETUP BUFFERS TO DO DMA READ
001057  030145  007000  000000        ADD     R0,UBAR,BAR                      ;ADD RING LENGTH TO BAR &
                                                                               ; LOAD RECIEVED DATA BUFFER TO RAM
001060  033741  000105  125640        MOV     #EDSEED,R1      %CALL   RD.DIAG ;SETUP EDC SEED AND
                                                                               ; GO READ DATA FROM HOST MEMORY.
001061  033440  010010  000043        MOV     R10,R0          %JNZRO  ERRB1B  ;ERROR IF NOT 0 /
                                                                               ; RESTORE RING LENGTH TO R0

                                        ;;Compare the EXPECTED & RECIEVED stored in RAM

001062  013440  000000  131104        NOP                     %CALL   S.BUFR  ;SETUP BUFFERS TO COMPARE DATA
001063  033702  010003  141046  5$:   MOV\F    (BUF),R2        %JZRO   3$      ;JUMP IF DONE, ELSE
                                                                               ; SAVE EXPECTED DATA.
001064  010145  007000  010000        ADD     R0,UBAR\N,BAR                    ;ADD RING LENGTH TO BAR &
                                                                               ; POINT TO RECIEVED DATA BUFFER IN RAM
001065  036502  000003  000000        XOR     (BUF),R2                         ;EXPECTED DATA SAME AS RECIEVED DATA?
001066  130445  017005  010043        INC     UBAR,,BAR       %JNZRO  ERRB1B  ;ERROR IF NOT THE SAME /
                                                                               ; INC TO NEXT RAM ADDRESS
001067  031450  000010  101063        DEC     R10             %JMP    5$      ;DECREMENT RING LENGTH.

001070  033751  006321  120253  6$:   MOV     #<B.NRTY+B.LOOP+B.LED+B.NABO>,R11,BREG %CALL GETID ;RELEASE D-PROC FROM SPECIAL WAIT LOOP
                                                                                                   ; CALLED 'TSTXL'. /
                                                                                                   ; PUT CONTROLLER IN LOOPBACK MODE
                                                                                                   ; TO READ CORRECT NODE ID
001071  013740  006221  100252        MOV     #<B.NRTY+B.LED+B.NABO>,BREG %JMP TSTRTN          ;PUT CONTROLLER IN NORMAL MODE /
                                                                                                   ; TEST UER & RETURN
```

```
                                ;;*****************************************************************************
                                ;
                                ;       INITIALIZATION SUBROUTINES
                                ;       THESE ROUTINES ARE USED BY THE INITIALIZATION PROCESS
                                ;
                                ;;*****************************************************************************

                                ;
                                ;       SUBROUTINE TO PRESET THE SDI CONTROL BLOCKS
                                ;

001072  033743  000004  125522  S.SSDI: MOV     #NSDI,R3          %CALL   S.SDI1  ;R3:# OF SDI CONTROL BLOCKS
                                                                                  ; UBAR (R5):1ST SDI CONTROL BLOCK PTR
001073  010545  017000  137777  SSSDIA: ADD     #SDI.ST,UBAR\N,BAR %RZRO          ;BAR:PTR TO SDI STATUS
001074  113740  003174  000000          MOV     #<ERRIP+DRVOL+DRAVL+SLAT+DRDUP>,\N,BUF ;INDICATE DRIVE OFFLINE
001075  030545  000120  000000          ADD     #SDIB.L,UBAR                      ;UPDATE UBAR (R5)
001076  031443  000003  111073          DEC     R3                %JMP    SSSDIA  ;DECR LOOP COUNTER

                                ;;*****************************************************************************
                                ; DMA ROUTINES
                                ;
                                ; This routine will write data to RAM, which will be used to write the ring
                                ; area in host memory.
                                ;
                                ;       Input:
                                ;               R7 = DATA
                                ;       Output:
                                ;               WRITE DATA TO RAM
                                ;               R2 = TEMPORARY DATA REGISTER
                                ;               R10 = RING LENGTH; WILL = 0 WHEN DONE
                                ;
                                ;;*****************************************************************************

001077  033442  000007  121104  WT.RAM: MOV     R7,R2             %CALL   S.BUFR  ;MOVE R7 DATA TO R2 / SETUP BUFFERS
001100  033442  013002  151104  1$:     MOV\F   R2,,BUF           %JZRO   S.BUFR  ;RETURN IF DONE, ELSE
                                                                                  ; STORE DATA IN RAM BUFFER.
001101  130445  007005  010000          INC     UBAR,,BAR                         ;INCREMENT TO NEXT RAM ADDRESS
001102  176542  000200  000000          XOR\L   #100000,R2                        ;CHANGE VALUE IN NEXT ADDRESS
001103  031450  000010  111100          DEC     R10               %JMP    1$      ;ALL DONE? LOOP
```



```
                                ;;*****************************************************************************
                                ;
                                ;       S.BUFR
                                ;
                                ; This routine will setup the ring length, the lo address bits of a ring area
                                ; in host memory and the starting address of a buffer in RAM.
                                ;
                                ;       Input:
                                ;               R0      = ring length
                                ;               TEMP3   = lo address bits of ring area in host memory
                                ;       Output:
                                ;               R10     = ring length
                                ;               IUAR    = lo address bits of ring area in host memory
                                ;               UBAR    = starting RAM buffer address for DMA xfer
                                ;               BAR     =       "                   "           "
                                ;
                                ;;*****************************************************************************

001104  033450  000000  000000  S.BUFR: MOV     R0,R10                            ;LOAD R10 WITH RING LENGTH.
001105  013740  007050  135600          MOV     #TEMP3,\N,BAR     %CALL   S.LDIU  ;LOAD IUAR (R6) WITH RING LO ADRS BITS.
001106  033745  000272  000000          MOV     #<BUFR2.>&LOBYT,UBAR              ;SET LO BITS &
001107  133545  007013  137777          BIS     #<BUFR2.>&HIBYT,UBAR,BAR %RET     ; HI BITS OF DATA BUFFER IN RAM
```

```
                                ;;*****************************************************************************
                                ;
                                ;       STEP.X
                                ;
                                ;       SEND DATA TO HOST DURING A STEP,
                                ;       RECEIVE DATA FROM HOST DURING STEP AND
                                ;       IN STEP 4, THE PROCESSOR WILL WAIT FOR THE GO BIT FROM HOST.
                                ;
                                ;       Input:
                                ;               R3 = STEP FLAG USED IN STEP
                                ;               Q = ECHO DATA
                                ;       Output:
                                ;               Q = DATA FROM THE HOST AND IS TESTED TO SET FLAGS ON RETURN
                                ;               R2 USED AS A TEMPORARY REG
                                ;               R1 USED AS A TEMPORARY REG
                                ;               R0 DESTROYED IN U.INTR
                                ;               R10 USED IN STEP
                                ;
                                ;       ROUTINES CALLED:
                                ;               DIINTR (CALLS U.INTR)
                                ;               STEP (LOADS SEND DATA INTO UDD [SA REG])
                                ;
                                ;;*****************************************************************************

001110  113740  004320  010000  SENDS2: MOV     #<LED8+LED4+LED1>,\N,UCRD       ;TURN ON STEP 2 LED.
001111  133743  000020  111115          MOV     #STEP2,R3       %JMP    STEP.X  ;SETUP STEP BIT / GO LOAD SA REG.

001112  113740  004300  000000  SENDS3: MOV     #<LED8+LED4>,\N,UCRD            ;TURN ON STEP 3 LEDS.
001113  133743  000040  111115          MOV     #STEP3,R3       %JMP    STEP.X  ;SETUP STEP BIT / GO LOAD SA REG.

001114  113740  004270  000000  SENDS4: MOV     #<LED8+LED2+LED1+DRINIT>,\N,UCRD ;TURN ON STEP 4 LED.

001115  033250  000010  010000  STEP.X: MOV     Q,UDDI                          ;LOAD UDDI SEND DATA FOR NEXT STEP
001116  033150  000003  000000          BIS     R3,UDDI                         ;SET STEP BIT FOR SA REG
001117  033441  000010  130365          MOV     UDDI,R1         %CALL   WRTSA   ;SEND STEP BIT TO HOST SA REG IN LOOPBACK MODE
001120  013740  006221  121134          MOV     #<B.NRTY+B.LED+B.NABO>,BREG %CALL DIINTR ;BI BACK TO NORMAL MODE /
                                                                                ; GO GENERATE AN INTERRUPT IF REQUIRED
001121  003443  000003  130246  1$:     MOV     R3,TSTCNT       %CALL   WAITS   ;LOAD CURRENT STEP /
                                                                                ; WAIT BEFORE ACCESSING BI AGAIN
001122  033740  000054  130313          MOV     #BIWSTA,R0      %CALL   RDCMD   ;READ BI WRITE STATUS REGISTER
001123  114542  000040  010000          BIT     #<GPR1>,R2                      ;WAS SA REG WRITTEN BY HOST?
001124  013440  010000  101121          NOP                     %JZR0   1$      ;WAIT TO RECEIVE DATA FROM HOST
                                                                                ;RESET GPR1 STATUS BIT
001125  033740  000054  130302          MOV     #BIWSTA,R0      %CALL   WRCMD   ;WRITE BI WRITE STATUS REGISTER
001126  033740  000364  130313          MOV     #SAREG,R0       %CALL   RDCMD   ;READ SA REG (BI GPR 1)
001127  003442  000002  010000          MOV     R2,Q                            ;GET DATA FROM SA REG
                                                                                ; LOAD DATA INTO Q REGISTER.
001130  114543  000100  000000          BIT     #STEP4,R3                       ;SEE IF WE ARE IN STEP 4,
001131  001240  010000  167777          DEC\F   Q               %RZR0           ;RETURN IF NOT, ELSE
                                                                                ; DECREMENT Q REG.
001132  013740  047170  051121          MOV\F   #CON.ST,\N,BAR  %JLSB   1$      ;JUMP IF GO BIT NOT SET, ELSE
                                                                                ;LOAD CONTROLLER STATUS POINTER TO BAR.
001133  033607  000003  127777          BIS     (BUF),Q,R7      %RET            ;R7 GETS Purge Interrupt BIT &
```



```
                                                                                ; STEP 4 RECEIVE DATA.

                                ;;*****************************************************************************
                                ; LEVO1
                                ;       DIINTR
                                ;               DO AN INTERRUPT IN DIAGNOSTIC
                                ;
                                ;       Input:
                                ;               R5 has the IE bit & interrupt vector address/4.
                                ;       Output:
                                ;               If U.INTR is called, Q, R0, & R1 are altered.
                                ;
                                ;;*****************************************************************************

001134  014545  000200  010000  DIINTR: BIT     #IE,R5                          ;IS Interrupt Enable SET?
001135  013440  010000  137777          NOP                     %RZR0           ;RETURN IF NOT, ELSE
001136  013440  000000  102030          NOP                     %JMP    U.INTR  ;DO INTERRUPT &
                                                                                ; DO RETURN IN U.INTR ROUTINE
```

```
                                ;;******************************************************************************
                                ;
                                ; LEV01
                                ;       SETLIM
                                ;               SET VALUES FOR LIMITS FOR THE 2 WORDS (4 bytes) THAT PRECEDE
                                ;               THE RING AREA FOR ALL SYSTEMS or  FOR THE ADDITIONAL 2 WORDS
                                ;               (4 bytes) FOR A SYSTEM WITH VAX PURGE SET.
                                ;
                                ;               THIS ROUTINE WILL CALL THE ERROR ROUTINE IF THE 32 BIT ADDRESS THAT
                                ;               MAKES UP THE PHYSICAL ADDRESS IN HOST MEMORY:
                                ;
                                ;                       1) IS < 4 FOR ALL SYSTEMS or < 8 FOR SYSTEMS WITH VAX PURGE SET
                                ;                       2) IS LOCATED AT A HIGH PHYSICAL ADDRESS & THE RING LENGTH IS
                                ;                          TOO LARGE FOR HOST MEMORY.
                                ;                       3) IS LOCATED IN THE SYSTEM I/O PAGE (last 4k of memory)
                                ;
                                ;               REGISTERS USED:
                                ;               Input:
                                ;                       Q  has hi order word of the physical address [ringbase+0] or
                                ;                          [ringbase-4] if VAX Purge Interrupt bit is set.
                                ;                       R0 has length of the ring
                                ;                       R2 has lo order word of the physical address [ringbase+0] or
                                ;                          [ringbase-4] if VAX Purge Interrupt bit is set.
                                ;               Output:
                                ;                       Q may need to be Decremented.
                                ;                       R0 will be Incremented by 2.
                                ;                       R2 gets lo order word of the physical address [ringbase-4] or
                                ;                          [ringbase-8] if VAX Purge Interrupt bit is set.
                                ;                       RAM location SAVUAR gets HI WORD of the physical address
                                ;                          [ringbase-4] or [ringbase-8] if VAX Purge Interrupt is set.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       122240 - DMA test error
                                ;
                                ;;******************************************************************************

001137  105640  000300  010000  SETLIM: AND     #BA,Q                           ;SAVE EXTENDED ADDRESS BITS
001140  131542  000004  000000          SUB     #4,R2                           ;DECREMENT LO ADDRESS BITS,
                                                                                ; WILL RING ADDRESS FIT IN LO ADDRESS SPACE?
001141  001260  020000  051142          DEC\T   Q               %TNCRY          ;CONTINUE IF SO, ELSE
                                                                                ; DECREMENT HI ADDRESS BITS,
                                                                                ; ARE WE SOMEWHERE IN HI MEMORY?
001142  133743  020340  040031          MOV\F   #160000,R3      %JNCRY  ERRB10  ;ERROR IF RING ADDRESS TOO LO OR HI IN MEMORY, ELSE
                                                                                ; SETUP TO CHECK FOR RING ADDRESS IN I/O PAGE
001143  117640  000300  010000          XOR     #BA,Q\N                         ;ARE ALL THE EXTENDED ADRS BITS SET ?
001144  035163  010002  151145          BIC\T   R2,R3           %TZR0           ;CONTINUE IF NOT, ELSE
                                                                                ; IS RING ADDRESS IN SYSTEM I/O PAGE?
001145  030540  010002  100031          ADD     #2,R0           %JZR0   ERRB10  ;ERROR IF RING ADDRESS IS IN SYSTEM I/O PAGE /
                                                                                ; ADD 2 TO RING LENGTH.
001146  013740  007262  115613          MOV     #SAVUAR,\N,BAR  %JMP    S.STQQ  ;STORE Q (Extended adrs bits) IN A
                                                                                ; RAM LOCATION FOR LATER & RETURN.
```



```
                                ;;******************************************************************************
                                ;
                                ; LEV00
                                ;       STEP 1 INIT CODE
                                ;
                                ;       DATA RECEIVED FROM HOST
                                ;               BIT15 is always SET
                                ;               Wrap mode bit (BIT14)
                                ;               Command ring length (BIT13-11)
                                ;               Response ring length (BIT10-08)
                                ;               Interrupt Enable bit (BIT07)
                                ;               Interrupt vector address/4 (BIT06-00)
                                ;
                                ;       REGS USED:
                                ;       Input:
                                ;               Q has the RECEIVE data from STEP 1.
                                ;       Output:
                                ;               Q gets SEND(ECHO) data for STEP 2.
                                ;               R5 gets SEND(ECHO) data for STEP 3.
                                ;               RAM location RSPLEN gets RESPONSE ring length.
                                ;               RAM location CMDLEN gets COMMAND ring length.
                                ;
                                ;       CALLS
                                ;               S.STR2
                                ;               S.LDR2
                                ;               S.RRR1
                                ;               S.RLR1
                                ;               S.RRR7
                                ;               S.STR0
                                ;
                                ; NOTE: If WRAP data is set, U-PROC will loop until reinited.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106200 - STEP 1 data error
                                ;
                                ;;******************************************************************************
                                STEP.1:
001147  033740  000054  130302          MOV     #BIWSTA,R0      %CALL   WRCMD   ;RESET GPR1 STATUS BIT
001150  033740  000364  130313          MOV     #SAREG,R0       %CALL   RDCMD   ;WRITE BI WRITE STATUS REGISTER
001151  003442  000002  010000          MOV     R2,Q                            ;READ SA REG (BI GPR 1)
001152  114640  030100  010034          BIT     #BIT14,Q        %JNNEG  ERRB1L  ;GET DATA FROM SA REG
                                                                                ;ERROR IF NEG NOT SET, /
                                                                                ; WRAP REQUESTED?
001153  013740  016221  111165          MOV     #<B.NRTY+B.LED+B.NABO>,BREG %JZR0 3$ ;JUMP IF NOT WRAP MODE /
                                                                                ; BI BACK TO NORMAL MODE
                                ;;Turn on LED #8, read SA REG source & write SA REG destination
001154  113740  004160  010000  1$:     MOV     #<LED4+LED2+LED1>,\N,UCRD       ;LED: 08H ON BOARD #1 /
001155  033740  000364  130313          MOV     #SAREG,R0       %CALL   RDCMD   ;READ SA REG (BI GPR 1)
001156  033441  000002  130365          MOV     R2,R1           %CALL   WRTSA   ;WRAP DATA FROM SA REG SOURCE TO SA REG DEST
                                                                                ; IN LOOPBACK MODE.
```

```
                                        ;;Wait for SA REG to be written again by HOST

001157  013440  000000  120246  2$:     NOP                     %CALL   WAITS   ;WAIT BEFORE ACCESSING BI AGAIN
001160  033740  000054  130313          MOV     #BIWSTA,RO      %CALL   RDCMD   ;READ BI WRITE STATUS REGISTER
001161  114542  000040  010000          BIT     #<GPR1>,R2                      ;WAS SA REG WRITTEN BY HOST?
001162  013740  016221  101157          MOV     #<B.NRTY+B.LED+B.NABO>,BREG %JZRO 2$ ;WAIT TO RECEIVE DATA FROM HOST /
                                                                                ; BI BACK TO NORMAL MODE
                                                                                ;RESET GPR1 STATUS BIT
001163  033740  000054  130302          MOV     #BIWSTA,RO      %CALL   WRCMD   ;WRITE BI WRITE STATUS REGISTER
001164  013440  000000  101154          NOP                     %JMP    1$      ;LOOP UNTIL REINITED

001165  033247  000007  121072  3$:     MOV     Q,R7            %CALL   S.SSDI  ;SAVE STEP 1 RECEIVED DATA IN R7,
                                                                                ; INIT EACH SDI STATUS TO REFLECT DRIVE OFFLINE
```



```
                                ;;************************ N   O   T   E   S ********************************
                                ;
                                ;       Make sure that the CLR HANG instruction and the CALL S.SSDI do not
                                ;       occur on the same line.  The D-PROC may not be released if they do.
                                ;       R5 must be NON-ZERO to continue.
                                ;
                                ;       Do not store anything in R7 until after the U-PROC has returned from
                                ;       S.SSDI. A NON-ZERO value in R7 will hang the D-PROC as long as R7 is
                                ;       the UER.
                                ;
                                ;       After this point the D.PROC is running and the U.PROC can use only
                                ;       registers Q, R0, R1, R2, R3, R5, R6, R7 and R10. The D.PROC owns R11,
                                ;       R12, R13, R14, R15, R16 and R17 from now on.
                                ;
                                ;       U.PROC will use R4 (CRI) if an error occurs.
                                ;
                                ;;*****************************************************************************

001166  034450  000010  000000          CLR     R10                             ;CLEAR R10 / USED TO STORE LENGTH
001167  013740  007020  000000          MOV     #RSPLEN,\N,BAR                  ;BAR -> RESPONSE RING LENGTH
001170  003740  000010  131176          MOV     #8.,Q           %CALL   GRNGLN  ;GET RING LENGTH / Q = # OF BITS TO SHIFT
001171  013740  007026  000000          MOV     #CMDLEN,\N,BAR                  ;BAR -> COMMAND RING LENGTH
001172  003740  000013  131176          MOV     #11.,Q          %CALL   GRNGLN  ;GET RING LENGTH
001173  033445  000007  000000          MOV     R7,R5                           ;R5 GETS STEP 3 SEND(ECHO) DATA
                                                                                ; (IE & INTERRUPT VECTOR/4).
001174  003740  000010  125503          MOV     #8.,Q           %CALL   S.RRR7  ;SWAB R7 FOR ECHO VALUE
001175  004547  000377  101202          AND     #377,R7,Q       %JMP    STEP.2  ;STRIP OFF GARBAGE, Q GETS STEP 2 SEND(ECHO) DATA &
                                                                                ; JUMP TO STEP 2.

                                ;;*****************************************************************************
                                ;
                                ;       GRNGLN
                                ;               GET RING LENGTH
                                ;               ISOLATE THE RING LENGTH LOG 2 AND GET THE ACTUAL RING LENGTH
                                ;
                                ;       Input:
                                ;               Q has # of bits.
                                ;               R7 has RECEIVE data from STEP 1.
                                ;               R10 has total ring length.
                                ;               BAR has some RAM location.
                                ;       Output:
                                ;               R1 is used in S.RRR1.
                                ;               R1 gets loaded into a RAM location.
                                ;               R10 gets ring length added to it.
                                ;
                                ;;*****************************************************************************

001176  037441  000007  135501  GRNGLN: COM     R7,R1           %CALL   S.RRR1  ;R1 GETS COMPLEMENTED DATA
001177  004541  000007  010000          AND     #7,R1,Q                         ;ISOLATE THE RING, STORE IN Q
001200  033741  010200  025501          MOV     #BIT07,R1       %CNZRO  S.RRR1  ;INITIAL VALUE IN R1 /
                                                                                ; CALL ONLY IF RING VALUE IS > 7
001201  020150  003001  127777          ADD     R1\0,R10,BUF    %RET            ;STORE IN BUFFER / ADD TO TOTAL LENGTH
```

```
                                ;;*****************************************************************************
                                ;
                                ; LEVOO
                                ;       STEP 2 INIT CODE
                                ;
                                ;       DATA SENT TO HOST
                                ;               Step 2 bit (BIT12)
                                ;               Port type (BIT10-08)
                                ;               Command ring length echo (BIT05-03)
                                ;               Response ring length echo (BIT02-00)
                                ;
                                ;       DATA RECEIVED FROM HOST
                                ;               Lo order word of the physical address [ringbase+0] (BIT15-01)
                                ;               VAX Purge Interrupt bit (BIT00)
                                ;
                                ;       REGS USED:
                                ;       Input:
                                ;               Q  has the SEND(ECHO) data for STEP 2.
                                ;               R5 has the SEND(ECHO) data for STEP 3.
                                ;       Output:
                                ;               Q gets the RECEIVE data from STEP 2.
                                ;               R0 used in U.INTR routine.
                                ;               R5 Still has the SEND(ECHO) data for STEP 3.
                                ;               RAM location INTVEC gets interrupt vector address/4.
                                ;               RAM location RCSRW1 (temp storage) gets total ring length.
                                ;               RAM location CON.ST gets VAX Purge Interrupt bit.
                                ;               RAM location RNGBSL gets LO WORD of the physical
                                ;                   address [ringbase+0].
                                ;
                                ;       Calls DIINTR for interrupts if requested.
                                ;
                                ;;*****************************************************************************

001202  034545  000377  010000  STEP.2: AND     #377,R5                         ;SAVE IE & INTERRUPT VECTOR
001203  013740  007164  010000          MOV     #INTVEC,\N,BAR                  ;BAR=INTERRUPT VECTOR STORAGE LOC
001204  014545  003177  010000          AND     #177,R5\N,BUF                   ;STORE INTERRUPT VECTOR/4
001205  073440  000010  010000          MOV\L   R10,R0                          ;CHANGE RING LENGTH TO WORDS
001206  013740  007025  000000          MOV     #RCSRW1,\N,BAR                  ;POINT TO TEMP RING LENGTH STORAGE IN RAM
001207  033440  003000  121110          MOV     R0,,BUF         %CALL   SENDS2  ;SAVE RING LENGTH (WORDS) AND
                                                                                ; CALL STEP#2 SEND ROUTINE.
001210  013740  007170  010000          MOV     #CON.ST,\N,BAR                  ;LOAD CONTROLLER STATUS POINTER TO BAR.
001211  014640  003001  010000          AND     #BIT00,Q\N,BUF                  ;SAVE VAX Purge Interrupt BIT IN CON.ST
001212  013740  007160  000000          MOV     #RNGBSL,\N,BAR                  ;POINTER -> LO RNG BASE ADDR
001213  005640  003001  010000          BIC     #BIT00,Q,BUF                    ;STORE LO ADDRESS IN RAM
```



```
                                ;;*****************************************************************************
                                ;
                                ; LEVOO
                                ;       STEP 3 INIT CODE
                                ;
                                ;       DATA SENT TO HOST
                                ;               Step 3 bit (BIT13)
                                ;               Interrupt Enable bit echo (BIT07)
                                ;               Interrupt vector address/4 echo (BIT06-00)
                                ;
                                ;       DATA RECEIVED FROM HOST
                                ;               Purge/Poll test bit (BIT15)
                                ;               Hi order word of the physical address [ringbase+0] (BIT14-00)
                                ;
                                ;       REGS USED:
                                ;       Input:
                                ;               R5 has the SEND(ECHO) data for STEP 3.
                                ;               R10 has the ring length.
                                ;               RAM location CON.ST has VAX Purge Interrupt bit.
                                ;       Output:
                                ;               Q gets Extended Address bits for the routine SETLIM.
                                ;               RAM location TEMP2 gets R5 [the SEND(ECHO) data for STEP 3].
                                ;               RAM location RNGBSH gets HI WORD of the physical
                                ;                   address [ringbase+0].
                                ;               RAM location TEMP3 gets LO WORD of the physical address
                                ;                   [ringbase-4] or [ringbase-8] if PI is set.
                                ;
                                ;       CALLS   SETLIM
                                ;               S.STR1
                                ;               UNB.RS
                                ;               SETLIM, WHICH SAVES THE EXTENDED ADDRESS BITS
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       122240 - DMA test error
                                ;
                                ;;*****************************************************************************

001214  003445  000005  131112  STEP.3: MOV     R5,Q            %CALL   SENDS3  ;RESTORE SEND(ECHO) DATA TO Q REG AND
                                                                                ; CALL STEP#3 SEND ROUTINE.
001215  013740  007047  125623          MOV     #TEMP2,\N,BAR   %CALL   S.STR5  ;SAVE R5, IT WILL BE DESTROYED
                                                                                ; DURING THE DMA TEST.
001216  013740  007162  125613          MOV     #RNGBSH,\N,BAR  %CALL   S.STQQ  ;SAVE HIGH BASE ADDRESS
001217  013440  030000  011226          NOP                     %JNNEG  1$      ;SKIP PURGE/POLL TEST IF PP BIT ISN'T SET /
                                                                                ; BAR -> RING BASE LO ADDR
001220  034441  000001  120365          CLR     R1              %CALL   WRTSA           ;CLEAR SA REG FOR PURGE/POLL TEST IN
                                                                                ; LOOPBACK MODE.
001221  013740  006221  130243          MOV     #<B.NRTY+B.LED+B.NABO>,BREG %CALL WAIT ;WASTE 10 MILLISECONDS /
                                                                                ; BI BACK TO NORMAL MODE
                                                                                ;THE SCAN/PURGE SIGNALS SHOULD HAVE BEEN
001222  013440  150000  110031          NOP                     %JNPOLL ERRB10  ; SET BY THE HOST AT THIS TIME.
                                                                                ;ERROR IF POLL WASN'T SET
001223  013740  116220  056237          MOV\F   #<B.NRTY+B.LED>,BREG    %JSTOP  BI.STP  ;JUMP IF BI STOPPED, ELSE       [mr1001]
                                                                                ; ABORT BI CMD
001224  013740  006221  130275          MOV     #<B.NRTY+B.LED+B.NABO>,BREG %CALL WASTE ;LOAD BREG WITH NO ABORT CMD /
```

```
                                                                        ; WASTE SOME TIME TO ALLOW
                                                                        ; RESET OF POLL SIGNAL.
001225  013440  150000  000031          NOP                     %JPOLL  ERRB10  ;ERROR IF POLL WASN'T RESET

001226  013740  007025  135573  1$:     MOV     #RCSRW1,\N,BAR  %CALL   S.LDRO  ;RESTORE RING LENGTH TO R0
001227  013740  007160  000000          MOV     #RNGBSL,\N,BAR                  ;BAR -> RING BASE LO ADDR
001230  033702  000003  121137          MOV     (BUF),R2        %CALL   SETLIM  ;LOAD R2 WITH RINGBASE+0 /
                                                                        ; SET LIMITS, R2 GETS RINGBASE-4
001231  013740  007170  125601          MOV     #CON.ST,\N,BAR  %CALL   S.LDR7  ;LOAD CONTROLLER STATUS POINTER TO BAR /
                                                                        ; R7 GETS Purge Interrupt BIT.
001232  034447  040007  031137          CLR     R7              %CLSB   SETLIM  ;R2 GETS RINGBASE-8 IF PI BIT IS SET /
                                                                        ; INIT R7 USED IN DMA TEST.
001233  013740  007050  135621          MOV     #TEMP3,\N,BAR   %CALL   S.STR2  ;STARTING ADDRESS OF RINGS,
                                                                        ; [LO WORD of the physical address [ringbase-4] or
                                                                        ; [ringbase-8] if VAX Purge Interrupt is set
```



```
                        ;;*****************************************************************************
                        ;
                        ; LEV00
                        ;       DMA TEST
                        ;
                        ;       For each location in the ring buffer do the following:
                        ;
                        ;               Write the moving ONE'S and ZERO'S patterns
                        ;               Clear the location
                        ;               Check to make sure the increment works properly
                        ;
                        ;       Input:
                        ;               R0 has length of ring area
                        ;               RAM location UBURST gets initialized to 2 words for burst rate
                        ;               RAM location TEMP3 has LO WORD of the physical address
                        ;                [ringbase-4] or [ringbase-8] if VAX Purge Interrupt is set
                        ;               RAM location SAVUAR has HI WORD of the physical address
                        ;                [ringbase-4] or [ringbase-8] if VAX Purge Interrupt is set
                        ;       Output:
                        ;               UDR gets changed by WT.DIAG & RD.DIAG routines
                        ;               UAR     "               "               "
                        ;               RING AREA in host memory will be initialized to data (0)
                        ;
                        ;
                        ;       REGISTERS USED:
                        ;
                        ;               R1 used when comparing read/write data
                        ;               R7 starting data pattern
                        ;               R10 used as temporary storage for ring length
                        ;               Q, R2, R3 are used in WT.DIAG & RD.DIAG routines
                        ;
                        ;               BUFR2. is a buffer in RAM that is used to store data
                        ;                       that will be written to the ring area in host memory.
                        ;               BUFR2. +516. is a buffer in RAM that is used to store data
                        ;                       that will be read from the ring area in host memory.
                        ;
                        ;       ROUTINES USED:
                        ;
                        ;               WT.RAM
                        ;               S.BUFR
                        ;               WT.DIAG
                        ;               RD.DIAG
                        ;               S.ZBUF
                        ;
                        ; NOTE: Errors in this routine display the following in the SA register;
                        ;       122240 - DMA test error
                        ;
                        ;;*****************************************************************************
```

```
001234  176547  000200  000000  DMATST: XOR\L   #100000,R7                          ;R7 = DATA
001235  013740  017140  111253          MOV     #UBURST,\N,BAR  %JZRO   SFT134  ;JUMP IF ZERO.

                                        ;;Setup to do DMA WRITE from a RAM buffer to host memory

001236  013740  003002  121077          MOV     #2,\N,BUF       %CALL   WT.RAM  ;GO WRITE DATA IN RAM BUFFER
001237  033741  000105  136245          MOV     #EDSEED,R1      %CALL   WT.DIAG ;SETUP EDC SEED AND
                                                                                ; GO WRITE DATA TO HOST MEMORY.
001240  033440  010010  000031          MOV     R10,R0          %JNZRO  ERRB10  ;ERROR IF NOT 0 /
                                                                                ; RESTORE RING LENGTH TO R0

                                        ;;Setup to do DMA READ from host memory to a RAM buffer

001241  013440  000000  131104          NOP                             %CALL   S.BUFR  ;SETUP BUFFERS TO DO DMA READ
001242  030145  007000  000000          ADD     R0,UBAR,BAR                             ;ADD RING LENGTH TO BAR &
                                                                                ; LOAD RECIEVED DATA BUFFER TO RAM
001243  033741  000105  125640          MOV     #EDSEED,R1      %CALL   RD.DIAG ;SETUP EDC SEED AND
                                                                                ; GO READ DATA FROM HOST MEMORY.
001244  033440  010010  000031          MOV     R10,R0          %JNZRO  ERRB10  ;ERROR IF NOT 0 /
                                                                                ; RESTORE RING LENGTH TO R0

                                        ;;Compare the EXPECTED & RECIEVED data from DMA xfer

001245  013440  000000  131104          NOP                             %CALL   S.BUFR  ;SETUP BUFFERS TO COMPARE DATA
001246  033702  010003  151234  1$:     MOV\F   (BUF),R2        %JZRO   DMATST  ;JUMP IF DONE, ELSE
                                                                                ; SAVE EXPECTED DATA.
001247  010145  007000  010000          ADD     R0,UBAR\N,BAR                   ;ADD RING LENGTH TO BAR &
                                                                                ; POINT TO RECIEVED DATA BUFFER IN RAM
001250  036502  000003  000000          XOR     (BUF),R2                        ;EXPECTED DATA SAME AS RECIEVED DATA?
001251  130445  017005  010031          INC     UBAR,,BAR       %JNZRO  ERRB10  ;ERROR IF NOT THE SAME /
                                                                                ; INC TO NEXT RAM ADDRESS
001252  031450  000010  101246          DEC     R10             %JMP    1$      ;DECREMENT RING LENGTH.
```



```
                                        ;;Clear RAM buffers & initialize RING area in host memory

001253  013440  000000  131104  SFT134: NOP                             %CALL   S.BUFR  ;SETUP BUFFERS TO CLEAR RAM BUFFERS.
001254  073440  000000  000000          ROT\L   R0                              ;RING LENGTH*2
001255  033441  000005  125544          MOV     UBAR,R1         %CALL   S.ZBUF  ;POINT TO BEGINNING OF BUFFERS USED
                                                                                ; DMA TEST / ZERO THE BUFFERS
001256  033440  000010  131104          MOV     R10,R0          %CALL   S.BUFR  ;RESTORE RING LENGTH /
                                                                                ; SETUP BUFFERS TO DO DMA WRITE
001257  033741  000105  136245          MOV     #EDSEED,R1      %CALL   WT.DIAG ;SETUP EDC SEED /
                                                                                ; GO WRITE DATA (0) TO HOST MEMORY.
001260  034440  010000  000031          CLR     R0              %JNZRO  ERRB10  ;ERROR IF NOT 0, ELSE
                                                                                ; INIT R0 TO WRITE ECC DATA TO RAM

                                        ;;Fill RAM with ECC data

001261  033741  000001  010000          MOV     #1,R1                           ;FILL RAM WITH ECC DATA
001262  137745  030374  111273  1$:     COM     #176000,R5      %JNEG   3$
001263  113540  007070  135620          BIS     #ALGADR,R0\N,BAR %CALL  S.STR1
001264  113541  007074  135616          BIS     #LGADR,R1\N,BAR  %CALL  S.STR0
001265  073441  000001  000000          ROT\L   R1
001266  114541  000004  010000          BIT     #BIT10,R1
001267  036541  010011  141271          XOR\F   #11,R1          %JZRO   2$
001270  135541  000004  010000          BIC     #BIT10,R1
001271  130440  000000  000000  2$:     INC     R0
001272  112140  000005  101262          CMP     R5,R0           %JMP    1$
001273  013740  007276  125623  3$:     MOV     #LGCKSV,\N,BAR  %CALL   S.STR5 ;INITIALIZE LOG/ANTILOG CHECK
001274                                  ASSUME  STEP.4,EQ,.
```

```
                                ;;*****************************************************************************
                                ;
                                ; LEV00
                                ;       STEP 4 INIT CODE
                                ;
                                ;       DATA SENT TO HOST
                                ;               Step 4 bit (BIT14)
                                ;               Controller model (BIT07-04)
                                ;               Controller micro-code version (BIT03-00)
                                ;
                                ;       DATA RECEIVED FROM HOST
                                ;               Burst rate (BIT07-02)
                                ;               Last Fail bit (BIT01)
                                ;               GO bit (BIT00)
                                ;
                                ;       REGISTERS USED:
                                ;       Input:
                                ;               BAR is pointing to RAM location CON.ST
                                ;               RAM location TEMP2 has R5 [the SEND(ECHO) data from STEP 3]
                                ;               RAM location CON.ST has VAX Purge Interrupt bit
                                ;       Output:
                                ;               R0 is a temporary reg used in U.INTR
                                ;               RAM location CON.ST gets Last Fail bit & VAX Purge
                                ;                Interrupt bit
                                ;               RAM location UBURST gets burst rate
                                ;
                                ;;*****************************************************************************

001274  003740  000043  000000  STEP.4: MOV     #<MODNUM+<CODVER&17>>&LOBYT,Q   ;GET LO BITS (make sure that            [mjt E125]
                                                                                ;code version # doesn't go over 4 bit boundary  &[mjt E125]
001275  103640  000001  000000          BIS     #<MODNUM+CODVER>&HIBYT,Q        ; HI BITS OF CONTROLLER MODEL #/uCODE VERSION
001276  013740  007047  135577          MOV     #TEMP2,\N,BAR   %CALL   S.LDR5  ;RESTORE R5, IE & INTERRUPT VECTOR ADDRESS/4
001277  133743  000100  131114          MOV     #STEP4,R3       %CALL   SENDS4  ;SETUP STEP BIT FOR UDD (SA REG) AND
                                                                                ; CALL STEP 4 SEND ROUTINE.

001300  114640  000001  000000          BIT     #SF,Q                           ;SPECIAL FUNCTION MODE?
001301  113740  014360  151315          MOV\F   #LEDS,\N,UCRD   %JZRO   3$      ;JUMP IF NOT, ELSE
                                                                                ; ALL LEDS OFF.
001302  037743  000000  000000          COM     #0,R3                           ;SETUP HOST DATA
001303  035545  000200  121115          BIC     #IE,R5          %CALL   STEP.X  ;MAKE SURE NO INTERRUPT OCCURS /
                                                                                ; GO LOAD SA REG WITH SF DATA.
001304  133544  000202  000000          BIS     #PLOCK+INDIAG,RLL               ;UNLOCK DPROC+DIAG MODE.
001305  114544  000200  010000  1$:     BIT     #PLOCK,RLL                      ;TEST PLOCK BIT
001306  044441  010401  051305          CLR\RQF R1      @SUPF   %JNZRO  1$      ;WAIT FOR D-PROC TO UNLOCK, ELSE
                                                                                ; INIT R1 & GET RID OF GO BIT /
                                                                                ; SET LOOP-ON-TEST FLAG.
001307  004640  000037  000000          AND     #37,Q                           ;SAVE TEST+PROCESSOR BIT.
001310  043241  000001  000000          MOV\RQ  Q,R1                            ;WILL IT BE U-PROC OR D-PROC TEST?
001311  133544  040200  051313          BIS\F   #PLOCK,RLL      %JLSB   2$      ;JUMP IF U-PROC TEST, ELSE
                                                                                ; UNLOCK D-PROC.
001312  013440  000000  125473          NOP                     %CALL   P.LOCK  ;WAIT FOR D-PROC TO UNLOCK ME.
```



```
001313  133544  000200  010000  2$:     BIS     #PLOCK,RLL                      ;UNLOCK D-PROC
001314  010640  002026  010000          ADD     #TESTBL,Q\N,PAR                 ;Q HAS TEST NUMBER /
                                                                                ;FORCE PAR TO TEST TABLE
                                3$:
001315  054641  000374  000000          AND\R   #374,Q,R1                       ;ISOLATE / SAVE BURST INFO IN R1
001316  003740  000010  125503          MOV     #8.,Q           %CALL   S.RRR7  ;ROTATE FLAGS INTO UPPER BYTE
001317  114547  003003  000000          AND     #<LFAIL+VAX.PG>,R7\N,BUF        ;SAVE LFAIL & VAX PURGE
001320  013740  007140  010000          MOV     #UBURST,\N,BAR                  ;BAR=BURST SAVE AREA
001321  030541  000002  135620          ADD     #2,R1           %CALL   S.STR1  ;SAVE BURST RATE / START U.PROC

                                        ;;Clear SA REGISTER and enter BI NORMAL MODE

001322  034441  000001  120365          CLR     R1              %CALL   WRTSA   ;CLEAR SA REG IN NORMAL MODE
                                                                                ; SELF-TEST OK & NO ABORT CMD.
001323  013740  006221  010000          MOV     #<B.NRTY+B.LED+B.NABO>,BREG     ;LOAD BREG WITH NO RETRY,
001324  013440  000000  101500          NOP                     %JMP    U.STRT  ;JUMP TO FUNCTIONAL START-UP CODE.

001325                                  .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO SEQERR
                                        .ORG    1500
001500                                  ASSUME  U.STRT,EQ,.                     ;MAKE SURE U.STRT IS NEXT
```

```
                                .SBTTL  U.PROC IDLE LOOP, PORT ROUTINES, SEEK ORDERING

                                ; ROUTINE NAME:
                                ;       U.STRT
                                ;
                                ; FUNCTIONAL DECRIPTION:
                                ;       THIS CODE PERFORMS THE UNIBUS PROCESSOR STARTUP FUNCTIONS
                                ; (AFTER POWERUP,BUS INIT,IP WRITE,OR RECEIPT COMMAND). THE
                                ; FUNCTIONS INCLUDE THE FOLLOWING:
                                ;       1. CLEAR ALL SDI STATUS WORDS
                                ;       2. SET UP SDI SYNC CODE
                                ;       3. CLEARS ALL SOFTWARE TIMER STORAGE
                                ;       4. CLEARS ALL SDI CONTROL BLOCKS
                                ;       5. SETS UP THE BUFFER STACK
                                ;
                                ;;-

001500  013740  007174  135631  U.STRT: MOV     #TMR.MC,BAR     %CALL   INIT1   ; [16K]SET MSCP TIMER FOR 1 SECOND
001501  013740  007176  122027          MOV     #TMR.BS,BAR     %CALL   U.VAXR  ; [16K]BAR=PTR TO TMR.BS/RESET VAX PURGE
001502  113740  003224  000000          MOV     #SECOND,BUF                     ; INIT MSCP TIMER
001503  013740  007200  000000          MOV     #HOSTMO,BAR                     ; [16K]BAR=PTR TO HOSTMO
001504  013740  003074  000000          MOV     #MINUTE,BUF                     ; SET HOST TIMER FOR 1 MINUTES
001505  013740  007202  010000          MOV     #SYNC,BAR                       ; [16K]BAR=PTR TO SYNC SAVE
001506  033741  000274  010000          MOV     #SYNCL,R1                       ; R1=LO BYTE OF SDI SYNC CHAR
001507  133541  003046  010000          BIS     #SYNCH,R1,BUF                   ; [16K]SET UPPER BYTE OF SDI SYNC
001510  013740  007142  000000          MOV     #CLIMIT,BAR                     ; [16K]BAR=COMMAND LIMIT WORD
001511  013740  003024  000000          MOV     #CMDLIM,BUF                     ; [16K]SET IN COMMAND LIMIT
001512  013740  007002  000000          MOV     #DMREG2,BAR                     ; [16K]BAR=PTR TO DM REG 2(SAVES SECTOR SIZE
001513  103740  003001  010000          MOV     #SECSZ,Q,BUF                    ; [16K]STORE SECTOR SIZE IN BUF & Q
001514  013440  000000  131572          NOP                             %CALL   U.CINT  ; [QDA]INIT PTE CACHE POINTERS

                                ; ***   SET UP BUFFER POINTER STACK ***

001515  033741  007257  010000          MOV     #BUFP42,R1,BAR                  ; [16K][U52EC1]R1,BAR=PTR TO 1ST STK ENTRY
001516  133744  000200  000000          MOV     #PLOCK,RLL                      ; [16K]SET IN PLOCK
001517  033144  000001  010000          BIS     R1,RLL                          ; [16K]OR IN STACK POINTER
001520  033742  000051  135530          MOV     #NBUFR,R2       %CALL   S.FMB1  ; [16K]R2=# BUFFERS/RO=ADDR OF 1ST BUFR

001521  013440  003000  000000  USTRTA: MOV     RO,BUF                          ; [16K]STORE BUFFER ADDR IN STACK
001522  131541  007002  010000          SUB     #2,R1,BAR                       ; [16K]UPDATE R1 AND BAR
001523  030040  000000  000000          ADD     Q,RO                            ; [16K]ADD SECTOR SIZE TO RO
001524  031442  000002  000000          DEC     R2                              ; [16K]DECR BUFR CNTRL BLK CNTR
001525  030540  010015  011521          ADD     #<BUF.DL-400>,RO %JNZRO USTRTA  ; [16K]ADD DIFFERENCE FROM BUF.DL
001526  013740  007242  135616          MOV     #CMPBUF,BAR     %CALL   S.STRO  ; [cho4]USE THE 42ND BUFFER AS A DEDICATED COMPARE BUFFER

                                ; ***   CHECK FOR LAST FAILURE ERROR LOG DESIRED ***

001527  133740  000002  135522          MOV     #LFAIL,RO       %CALL   S.SDI1  ; UBAR=ADDRESS OF SDI.1
001530  013740  007170  135603          MOV     #CON.ST,BAR     %CALL   S.BITO  ; IF LAST FAILURE PKT NOT DESIRED
001531  013740  017034  101537          MOV     #FAILUR,BAR     %JZRO U.IDLE    ;   THEN SKIP                           [rae09]
001532  013700  000003  010000          TST     (BUF)                           ; IF LAST FAILURE = 0                   [rae09]
001533  010545  017010  111537          ADD     #SDI.1T,UBAR\N,BAR %JZRO U.IDLE ;   THEN DON'T SEND IT                 [rae09]
001534  013740  013000  101537          MOV     #FM.CNT,BUF     %JZRO   U.IDLE  ;   ELSE STORE LOG CODE IN SDI.1T       [rae09]
001535  010545  007003  000000          ADD     #SDI.SW,UBAR\N,BAR              ; BAR=PTR TO EVENT CODE
```



```
001536  013740  003012  133233          MOV     #ST.CNT,BUF     %CALL   U.LOGS  ; SET EVENT TO CONTROLLER ERROR
```

```
                                .SBTTL  U.PROC IDLE LOOP
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.IDLE (UNIBUS PROCESSOR IDLE LOOP)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS CODE IS THE MAIN SCHEDULING LOOP FOR THE UNIBUS PROCESSOR
                                ; AND AS SUCH PERFORMS THE FOLLOWING FUNCTIONS:
                                ;       1. SENDS CONTROLLER TO HOST COMMANDS TO THE HOST MESSAGE RING
                                ;       2. PERFORMS HOST COMMAND RING POLLING
                                ;       3. MAINTAINS THE CONTROLLER'S MASTER SOFTWARE COUNTER FOR MSCP/SDI TIMING
                                ;       4. MONITORS THE SDI STATUS WORDS FOR TASKS TO PERFORM ON BEHALF
                                ;          OF THE DRIVE PROCESSOR
                                ;
                                ; INPUTS:
                                ;       1. SDI CONTROL BLOCKS
                                ;       2. TMR.BS                       CONTROLLER INTERNAL 1 SECOND TIMER
                                ;
                                ; OUTPUTS:
                                ;       1. UPDATED SDI CONTROL BLOCKS
                                ;       2. TMR.BS                       CONTROLLER INTERNAL 1 SECOND TIMER
                                ;
                                ;;-

                                ; ***   CHECK FOR HOST POLL FLAG SET ***

001537  013740  157020  021675  U.IDLE: MOV     #RSPLEN,BAR     %CPOLL  U.RECV  ; IF POLL FLAG THEN GO CHECK COMMAND RING
001540  013740  007034  125630          MOV     #FAILUR,BAR     %CALL   S.CLRB  ; ZAP LAST FAILURE


                                ; ***   CHECK FOR D.PROC FATAL ERROR OR DMDT ENTRY REQUEST ***

001541  114544  050010  001624  U.RAMX: BIT     #CN.ERR,RLL     %JTEST  U.RAMD  ; [16K]IF FATAL CONTROLLER ERROR
001542  114544  010100  012066          BIT     #DMODE,RLL      %JNZRO  U.UERD  ; [16K]THEN GO TREAT IT
001543  033745  010355  051562          MOV\F   #SDI.1&LOBYT,UBAR %JNZRO U.IDLC ; [16K]IF IN DM MODE THEN SKIP CNTRL BLK CHK;[E121]
001544  133545  007002  010000          BIS     #SDI.1&HIBYT,UBAR,BAR           ; [16K]UBAR=ADDRESS OF SDI.1                    ;[E121]
                                                                                                                            ;[E121]
                                ; ***   PERFORM SEEK ORDERING/CHECK SDI STATUS WORDS FOR D.PROC REQUESTS ***             ;[E121]
                                ;       AT U.IDLB, BAR=UBAR                                                             ;[E121]
                                                                                                                        ;[E121]
001545                          U.IDLB: ASSUME  SDI.ST,EQ,0                                                             ;[E121]
001545  033701  000003  010000          MOV     (BUF),R1                        ; R1 = SDI CTL BLK STATUS               ;[E121]
001546  114541  000001  010000          BIT     #DATT,R1                        ; SEE IF LOWER REPORTED ATTN            ;[E121]
001547  010545  017054  111554          ADD     #<SDI.AT+P.OPCD>,UBAR\N,BAR %JZRO 10$   ; IF NOT, SKIP ATTN PROC       ;[E121]
001550  033700  000003  000000          MOV     (BUF),R0                        ; GET OPCODE WD OF INTERNAL MSCP PKT    ;[E121]
001551  033747  010017  011554          MOV     #OP.ATT,R7      %JNZRO  10$     ; IF SET UP, SKIP ATTN PROC             ;[E121]
001552  114541  000030  122313          BIT     #DRVOL+DRAVL,R1 %CALL   U.ATTN  ; QUEUE INTERNAL ATTN PACKET            ;[E121]
001553  010545  007000  125574          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR1 ; R1=SDI CONTROL BLK STATUS             ;[E121]
001554  013740  007030  125623  10$:    MOV     #CMDCOF,BAR     %CALL   S.STUB  ; SAVE UBAR                             ;[E121]
001555  014541  000001  133766          BIT     #PKIP,R1        %CALL   U.CSDI  ; GO CHECK IF PACKET IN PROGRESS        ;[E121]
001556  013740  007030  135577          MOV     #CMDCOF,BAR     %CALL   S.LDUB  ; RESTORE UBAR                          ;[E121]
001557  106545  000003  010000          XOR     #SDI.4&HIBYT,UBAR,Q             ; ARE WE AT THE LAST SDI CTL BLK?       ;[E121]
001560  006640  000335  000000          XOR     #SDI.4&LOBYT,Q                                                          ;[E121]
001561  030545  017120  001545          ADD     #SDIB.L,UBAR,BAR %JNZRO U.IDLB  ; IF NOT, LOOP 'TIL WE ARE              ;[E121]
```



```
                                ; ***   CHECK FOR RESPONSES TO SEND TO HOST ***

001562  013740  007204  125573  U.IDLC: MOV     #HSTQUE,BAR     %CALL   S.LDR0  ; R0=CONTROLLER->HOST QUE HEAD
001563  034442  010002  001627          CLR     R2              %JNZRO  U.SEND  ; IF NEQ 0 THEN PKTS TO SEND

                                ; ***   CHECK FOR DM XFC SERVICE REQUIRED ***

001564  114544  000020  135402  U.IDLD: BIT     #DXFC,RLL       %CALL   U.XFCP  ; IF D.PROC XFC SERVICE REQUIRED

                                ; ***   CHECK FOR ONE SECOND INTERVAL ***

001565  013740  007176  125576          MOV     #TMR.BS,BAR     %CALL   S.LDR3  ; R3=CONTROLLER TIMER                   ;[E121]
001566  031443  010003  001571          DEC     R3              %JNZRO  10$     ;                                       ;[E121]
001567  133743  000224  121603          MOV     #SECOND,R3      %CALL   U.TIMR  ; IF TIMER=0 THEN CHK SDI TIMERS        ;[E121]
001570  013740  007176  010000          MOV     #TMR.BS,BAR                     ; AND RESET TIMER                       ;[E121]
001571  013440  003003  111537  10$:    MOV     R3,BUF          %JMP    U.IDLE  ; STORE NEW TIMER VALUE & LOOP          ;[E121]
```

```
                                ;;+
                                ;ROUTINE NAME:
                                ;               U.CINT
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;               THIS ROUTINE SETS UP THE POINTER TO THE PTE CACHE IN EACH
                                ;               SDI CONTROL BLOCK.
                                ;;-

001572  033741  000317  000000  U.CINT: MOV     #STCACH&LOBYT,R1                ; [QDA]GET START OF CACHE AREA LO PORTION
001573  133541  000066  125522          BIS     #STCACH&HIBYT,R1 %CALL S.SDI1   ; [QDA]SET IN HI ADDRESS/UBAR = ADDR FIRST SDI CTRL BLK
001574  010545  007101  000000  U.CIN1: ADD     #MAP.CH,UBAR\N,BAR              ; [QDA]POINT TO CACHE PTR
001575  013440  003001  010000          MOV     R1,BUF                          ; [QDA]STORE PTR
001576  030541  000104  010000          ADD     #(MEMSZ*2),R1                   ; [QDA]POINT TO NEXT CACHE AREA
001577  030545  000120  135532          ADD     #SDIB.L,UBAR    %CALL S.FBC1    ; [QDA]POINT TO NEXT SDI CONTROL BLOCK
001600  036140  000005  010000          XOR     UBAR,R0                         ; [QDA]CHECK FOR END
001601  013440  010000  001574          NOP                     %JNZRO U.CIN1   ; [QDA]LOOP TILL DONE
001602  013440  000000  127777          NOP                     %RET            ; [QDA]RETURN
```



```
                                ;++
                                ; ROUTINE NAME:
                                ;       U.TIMR (U PROCESSOR TIMER ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL MAINTAIN AND UPDATE ACTIVE SDI CONTROL BLOCK
                                ; TIMERS AND THE MASTER MSCP TIMER.
                                ;
                                ; INPUTS:
                                ;       ENTERED EVERY 1 SECOND
                                ;
                                ; OUTPUTS:
                                ;       UPDATED SDI CONTROL BLOCK TIMERS
                                ;       UPDATED HOST MSCP TIMER'
                                ;
                                ;;-

                                ; ***   UPDATE SDI INTERCONNECT TIMERS IF ACTIVE ***

001603  033742  000004  135522  U.TIMR: MOV     #NSDI,R2        %CALL   S.SDI1  ; R2=SDI BLK CNT/UBAR=1ST SDI PTR
001604  003700  000014  000000          MOV     (UCRS),Q                        ;Q=LED REG
001605  005640  000377  125473          BIC     #377,Q          %CALL   P.LOCK  ;GET RID OF FLOATING BITS FROM UCRS /
                                                                                ; UBAR=PTR TO SDI CNTL BLK TIMER
001606  114544  000100  010000          BIT     #DMODE,RLL                      ; [16K]IF IN DM MODE
001607  106640  014020  015474          XOR     #LED1,Q,UCRD    %JNZRO  UNLOCK  ; [16K]  THEN FLIP/FLOP LED & RETURN    ;[E121]
001610  010545  007004  135614  U.TMRA: ADD     #SDI.TM,UBAR\N,BAR %CALL S.DECB ; BAR=UBAR/GO DECR TIMER
001611  031442  000002  000000          DEC     R2                              ; DECR COUNTER
001612  030545  010120  011610          ADD     #SDIB.L,UBAR    %JNZRO  U.TMRA  ; IF COUNTER NEQ 0 THEN LOOP
001613  133544  000200  010000          BIS     #PLOCK,RLL                      ; [16K]RELEASE D.PROC
001614  013740  007174  135573          MOV     #TMR.MC,BAR     %CALL   S.LDR0  ; BAR=MSCP TIMER PTR/R0=TIMER
001615  031460  010000  045616          DEC\T   R0              %JNZRO  S.STR0  ; IF TIMER NEQ 0 THEN DECR AND UPDATE
001616  033747  000011  131622          MOV     #ER.HTO,R7      %CALL   U.HTMO  ; R7=HOST TIMEOUT ERROR/RESET TMR.MC
001617  105660  014010  167777          BIC\T   #DRINIT,Q,UCRD  %RZRO           ; [16K]IF HOST TIMEOUT EQ 0 THEN DON'T
001620  114640  000010  000000          BIT     #DRINIT,Q                       ; [16K]IF DRINIT CLEAR
001621  105640  014010  102071          BIC     #DRINIT,Q,UCRD  %JZRO   U.UERR  ; [16K]  THEN FATAL ERROR AND GO AVAILABLE

                                ; ***   ROUTINE TO RESET HOST TIMEOUT (USES R1) ***

001622  013740  007200  125574  U.HTMO: MOV     #HOSTMO,BAR     %CALL   S.LDR1  ; R1=HOST TIME OUT (0=NO TIMEOUT)
001623  013740  007174  105620          MOV     #TMR.MC,BAR     %JMP    S.STR1  ; RESET HOST TIMER/RETURN

                                ; ***   RING POLL TIMER VALUE ***

112000                          SECOND  := 112000                               ; APPROXIMATE ONE SECOND VALUE          ;[E121]

                                ; ***   U.PROC RAM DUMP ROUTINE - ENTERED FROM U.RAMX WHEN "TEST" CONDITION IS TRUE
                                ; ***   THE ROUTINE WILL START AT RAM LOCATION ZERO AND LOAD EACH WORD OF THE 16K RAM

001624  034445  007005  010000  U.RAMD: CLR     UBAR,UBAR,BAR                   ; [16K]CLEAR UBAR, LOAD BAR
001625  013705  050003  111541  URAMDA: MOV     (BUF),UBAR\N    %JNTEST U.RAMX  ; [16K]PUT RAM CONTENTS ON BUS/JMP BACK IF DONE
001626  130445  007005  101625          INC     UBAR,UBAR,BAR   %JMP    URAMDA  ; [16K]INCR UBAR,LOAD BAR,LOOP
```

```
                                .SBTTL  UNIBUS/Q-BUS PORT ROUTINES

                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SEND (SEND CONTROLLER MSCP PACKET TO HOST RESPONSE RING)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL DO THE FOLLOWING:
                                ;       1. CALCULATE RESPONSE RING ADDRESS
                                ;       2. DETERMINE IF THE HOST HAS A RESPONSE BUFFER AVAILABLE
                                ;       3. IF SO, THEN TRANSFER THE MSCP PACKET TO THE HOST
                                ;       4. UPDATE THE HOST RESPONSE RING POINTER (IF TRANSFER DONE)
                                ;       NOTE THAT ALL OF THE POSSIBLE CONTROLLER TO HOST PACKETS ('END','LOG',
                                ; AND, 'ATTENTION') ARE SENT USING THIS MECHANISM,I.E.,
                                ; THE PACKET IS ENTERED ON THE CONTROLLER TO HOST QUEUE 'HSTQUE'.
                                ;
                                ; INPUTS:
                                ;       R2              = 0
                                ;       CRI             SYSTEM STATUS WORD
                                ;       HSTQUE          CONTROLLER -> HOST RESPONSE/MESSAGE QUEUE HEAD
                                ;
                                ; OUTPUTS:
                                ;       ALL U.PROC REGISTERS ARE USED
                                ;
                                ;;-

                                ; ***   CALCULATE HOST RESPONSE RING ADDRESS ***

001627  033745  000020  131737  U.SEND: MOV     #RSPLEN,UBAR    %CALL   U.RCSR  ; GO READ RESPONSE CSR REGS
001630                                  ASSUME  OWN,EQ,BIT15                    ; MAKE SURE OWN IS MSB
001630  013440  030000  001564          NOP                     %JNMSB  U.IDLD  ; IF NO BUFR THEN RETURN                ;[E121]

                                ; ***   READ LO CSR ENTRY FOR MSCP RESPONSE BUFFER ADDRESS ***

001631  033745  000022  121746          MOV     #RSPCOF,UBAR    %CALL   U.RCSB  ; [16K]GO GET BUFR ADDR IN R7
001632  033446  000007  000000          MOV     R7,IUAR                         ; [16K]IUAR=BUFR ADDR

                                ; ***   RESET MSCP HOST TIMER ***

001633  033701  000014  010000          MOV     (UCRS),R1                       ; [16K]R1=UCR DATA
001634  114541  000010  010000          BIT     #DRINIT,R1                      ; [16K]IF NOT WAITING FOR DRIVES
001635  033740  010060  131622          MOV     #MCP.LN*2,RO    %CZRO   U.HTMO  ;   THEN RESET MSCP HOST TIMER

                                ; ***   GET PACKET ADDRESS IN R10 ***

001636  013740  007204  125602          MOV     #HSTQUE,BAR     %CALL   S.LD10  ; R10=RESPONSE PACKET ADDRESS

                                ; ***   SEND CONTROLLER RESPONSE PACKET TO THE HOST ***

001637  003740  000004  000000          MOV     #4,Q                            ; [16K]Q=4
001640  013740  007025  132003          MOV     #RCSRW1,BAR     %CALL   U.SADX  ; [16K]BAR=PTR TO RSP CSRW1/GO ADJUST PKT ADDRESS

                                ; ***   DETERMINE PACKET LENGTH ***
```



```
001641  010550  007006  125575          ADD     #P.OPCD,R10\N,BAR %CALL S.LDR2  ; R2=END PACKET OP CODE
001642  014542  000060  010000          BIT     #IOMSK,R2                       ; IF I/O COMMAND
001643  033760  010040  051664          MOV\T   #MCP.IO*2,RO    %JNZRO  U.SNDB  ;   THEN WRITE SMALLER PACKET
001644  016550  000304  010000          XOR     #LOGPKT,R10\N                   ; IF PKT NEQ LOGPKT
001645  014542  010200  011664          BIT     #OP.END,R2      %JNZRO  U.SNDB  ; IF EQ LOG OR ATTN
001646  013740  017154  135573          MOV     #LOGLEN,BAR     %CZRO   S.LDRO  ;   THEN GET RO FROM LOGLEN
001647  033445  007010  135574  U.SNDA: MOV     R10,UBAR,BAR    %CALL   S.LDR1  ; R1=PKT STATUS & LINK WORD

                                ; ***   PUT LENGTH IN PACKET, SAVE STATUS AND LINK WORD ***

001650  013440  003000  000000          MOV     RO,BUF                          ; PUT LENGTH*2 IN PACKET(BYTES)
001651  013740  007152  125620          MOV     #QUESAV,BAR     %CALL   S.STR1  ; SAVE STATUS AND LINK WORD
001652  050540  000004  126242          ADD\R   #P*2,RO         %CALL   UNB.WR  ; GO SEND PACKET TO HOST
001653  033767  010002  052071          MOV\T   #ER.PWR,R7      %JNZRO  U.UERR  ; IF ERROR THEN FATAL

                                ; ***   DEQUEUE PACKET AND CHECK IF CONTROLLER -> HOST QUEUE EMPTY ***

001654  013740  007152  125573          MOV     #QUESAV,BAR     %CALL   S.LDRO  ; RO=PKT STATUS & LINK WORD
001655  033747  007204  135574          MOV     #HSTQUE,R7,BAR  %CALL   S.LDR1  ; R1=PKT STATUS & LINK WORD
001656  013440  007001  135616          MOV     R1,BAR          %CALL   S.STRO  ; RESTORE STATUS AND LINK WORD
001657  013440  007007  123713          MOV     R7,BAR          %CALL   U.ULNK  ; GET NEXT PKT FROM HOST QUEUE
001660  013740  017020  131675          MOV     #RSPLEN,BAR     %CZRO   U.RECV  ; IF EQ 0 THEN CLR SEND PKT FLAG & CHECK COMMAND RING

                                ; ***   UPDATE HOST CSR WORDS, UPDATE RING POINTER, CHECK FOR INTERRUPT  ***

001661  033745  000025  121755          MOV     #RCSRW1,UBAR    %CALL   U.UCSR  ; [16K]COMMON ROUTINE TO UPDATE RING CSR
001662                                  ASSUME  OWN,EQ,BIT15                    ; MAKE SURE OWN IS MSB
001662  003740  030002  122013          MOV     #RSPO,Q         %CMSB   U.INTI  ; IF OWN EQ 1 THEN GEN INTERRUPT
001663  013440  000000  111562          NOP                     %JMP    U.IDLC  ;   ELSE LOOK FOR MORE TO SEND

                                ; ***   SET CONTROLLER COMMAND LIMIT - INCREMENTALLY ***

001664  013740  007142  125574  U.SNDB: MOV     #CLIMIT,BAR     %CALL   S.LDR1  ; [16K]R1=COMMAND LIMIT
001665  033743  010016  111647          MOV     #14.,R3         %JZRO   U.SNDA  ; [16K]IF EQ 0 THEN CONTINUE
001666  112141  000003  010000          CMP     R3,R1                           ; [16K]IF R3(14) LT R1(CMD LIM)
001667  033463  020001  141670          MOV\T   R1,R3           %TCRY           ; [16K]   THEN R3=REMAINDER
001670  131141  000003  135620          SUB     R3,R1           %CALL   S.STR1  ; [16K]R1=REMNANT/RESET CLIMIT
001671  010550  007001  125574          ADD     #P.VCID,R10\N,BAR %CALL S.LDR1  ; R1=VIRTUAL CIRCUIT ID
001672  035541  000017  010000          BIC     #LONIB,R1                       ; ISOLATE LO NIBBLE
001673  130141  000003  125620          ADDC    R3,R1           %CALL   S.STR1  ; [16K]R1=R1+R3+1/RESET P.VCID
001674  013440  000000  101647          NOP                     %JMP    U.SNDA  ; [16K]CONTINUE
                                ;       OR      #CMDLIM+1,R1,BUF %JMP   U.SNDA  ; [16K]SET IN COMMAND LIMIT/CONTINUE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.RECV (RECEIVE MSCP COMMAND ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:
                                ;       1. RETURNS IF NO PACKETS ARE AVAILABLE
                                ;       2. FINISHES CALCULATING THE CURRENT RING ENTRY ADDRESS
                                ;       3. SETS IUAR AND CR WITH THE PROPER UNIBUS ADDRESS
                                ;       4. READS 1ST TWO WORDS OF THE CSR DESCRIPTOR INTO R7 AND R10
                                ;       4. CHECKS THE 'OWN' BIT OF THE CSR DESCRIPTOR
                                ;       5. IF 'OWN' IS NOT SET
                                ;               THEN RETURN WITH NO RING ACTION PERFORMED
                                ;               ELSE
                                ;               A. RESET THE MSCP ACTIVITY TIMER
                                ;               B. CLEAR THE MSCP TIMEOUT FLAG IN THE CONTROLLER STATUS WORD
                                ;               C. READ THE MSCP PACKET INTO THE BUFFER
                                ;               D. GO TO 1. ABOVE
                                ;
                                ; INPUTS:
                                ;       BAR = RSPLEN
                                ;
                                ; OUTPUTS:
                                ;
                                ;       PACKET READ INTO CONTROLLER RAM
                                ;
                                ;;-

                                ; ***   CALCULATE COMMAND RING ENTRY ADDRESS ***

001675  033702  000014  010000  U.RECV: MOV     (UCRS),R2                       ; [16K]R2=UCR DATA
001676  114542  000010  010000          BIT     #DRINIT,R2                      ; [16K]IF DRIVES NOT STILL INITING
001677  033722  010003  172005          MOV\T   (BUF),R2        %CZRO   U.GPKT  ;   THEN R2=RESPONSE LEN/GET PACKET
001700  070142  010002  027777          ADD\L   R2,R2           %RNZRO          ; IF NO PKT AVAILABLE THEN RETURN
001701  003740  116220  056235          MOV\F   #<B.NRTY+B.LED>,Q,BREG %JSTOP BI.STO    ; [BDA] CLR POLL BIT NOW TO     ;[E121]
001702  003640  006001  010000          BIS     #<B.NABO>,Q,BREG                ; [BDA] PREVENT CONTINUOUS POLLING      ;[E121]

                                ; ***   READ COMMAND HI CSR ENTRY ***

001703  033745  000026  131737          MOV     #CMDLEN,UBAR    %CALL   U.RCSR  ; [16K]GO READ HI CSR REGS TO R7        ;[E121]
001704                                  ASSUME  OWN,EQ,BIT15                    ; MAKE SURE OWN IS MSB

                                ; ***   RESET MSCP HOST TIMER ***

001704  013440  030000  037777          NOP                     %RNMSB          ; [BDA] MAKE SURE SA STILL ZERO

                                ; ***   READ LO CSR ENTRY FOR MSCP COMMAND ADDRESS ***

001705  033745  000030  121746          MOV     #CMDCOF,UBAR    %CALL   U.RCSB  ; [16K]GO GET CMD ADDR IN R7

                                ; ***   READ MSCP PACKET INTO FREE BUFFER ***

001706  033446  000007  132005          MOV     R7,IUAR         %CALL   U.GPKT  ; [16K]SET UP UNIBUS ADDR/GET PKT ADDR  ;[E121]
```



```
001707  113740  003200  131622          MOV     #PSTAT,BUF      %CALL   U.HTMO  ; INDICATE ENTRY IN USE/RESET HOST TIMEOUT

                                ; *** EXTRACT MSCP MSG LENGTH FROM PACKET HEADER ***                    [rae02]

001710  003740  000004  000000          MOV     #4,Q                            ; msglen adr = -4 bytes [rae02]
001711                                  ASSUME  UQ.CNT,EQ,0                     ; which is the offset of the length word;[E121]
001711  033445  000010  000000          MOV     R10,UBAR                        ; UBAR = PKT Adrs       [rae02]         ;[E121]
001712  013740  007033  122003          MOV     #CCSRW1,BAR     %CALL   U.SADX  ; BAR=CSRW1 adrs/adjust src adrs [rae02];[E121]
001713  033740  000024  135635          MOV     #MCP.RD+2,R0    %CALL   UNB.RD  ; read packet from host [rae02]         ;[E121]
001714  033767  010006  042071          MOV\T   #ER.RRD,R7      %JNZRO  U.UERR  ;  if error then FATAL  [mjt01]
001715  010550  007000  125573          ADD     #UQ.CNT,R10\N,BAR %CALL S.LDRO  ; get count             [rae02]
001716  013740  007264  125616          MOV     #MSCPLN,BAR     %CALL   S.STRO  ; and save it           [rae02]

                                ; ***   Check the msg length for minimum size                           [rae02]

001717  131540  000014  000000          SUB     #14,R0                          ; must be at least 14   [rae02]
001720  034441  030001  111735          CLR     R1              %JNEG   U.RCVE  ;  if not then error   [rae02]

                                ; ***   WRITE UPDATED CSR AND CHECK IF INTERRUPT NEEDED ***

001721  013740  007046  135626          MOV     #TEMP1,BAR      %CALL   S.ST10  ; SAVE R10 AROUND CALL TO U.UCSR
001722  033745  000033  131755          MOV     #CCSRW1,UBAR    %CALL   U.UCSR  ; [16K]USE COMMON ROUTINE TO UPDATE CSR
001723                                  ASSUME  OWN,EQ,BIT15                    ; MAKE SURE OWN IS MSB
001723  003740  030004  122013          MOV     #CMDQ,Q         %CMSB   U.INTI  ; SEND INTERRUPT TO HOST IF REQUIRED
001724  013740  007046  135602          MOV     #TEMP1,BAR      %CALL   S.LD10  ; RESTORE PACKET ADDRESS TO R10

                                ; ***   DO PRELIMINARY PROCESSING OF MSCP PACKET ***
                                ; ***   Verify the reserved fields common to all opcodes ***

001725  010550  007005  122240          ADD     #P.RS03,R10\N,BAR %CALL  S.TST  ; test reserved word 3          [rae02]
001726  033761  010003  051735          MOV\T   #<P.RS03-P>,R1    %JNZRO U.RCVE ; if bad declare in error       [rae02]
001727  010550  007008  125573          ADD     #P.OPCD,R10\N,BAR %CALL  S.LDRO ; test hi byte of op code       [rae02]
001730  015540  000377  010000          BIC     #LOBYT,R0\N                     ;                               [rae02]
001731  033761  010004  041735          MOV\T   #<P.OPCD-P>,R1    %JNZRO U.RCVE ; if bad declare in error       [rae02]

001732  010550  007001  135573          ADD     #P.VCID,R10\N,BAR %CALL S.LDRO  ; [16K]R0=VIRT CIR ID
001733  035540  003017  122166          BIC     #LONIB,R0,BUF   %CALL   U.OPCD  ; [16K]ZAP CREDIT FIELD/GO CHECK OP CODE
001734  013740  007020  111675          MOV     #RSPLEN,BAR     %JMP    U.RECV  ; GO CHECK NEXT CSR
                                                                                ; error detected in MSCP format [rae02]
                                                                                ; reset UPF becuase U.OPCD does [rae02]
001735  013440  000600  132506  U.RCVE: NOP             @RUPF   %CALL   U.CMDA  ; go process error             [rae02]
001736  013740  007020  111675          MOV     #RSPLEN,BAR     %JMP    U.RECV  ; GO CHECK NEXT CSR             [rae02]
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.RCSR (READ CSR REGISTERS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL READ THE CSR REGISTER POINTED TO                       ;[E121]
                                ;       BY IUAR INTO THE WORD POINTED TO BY UBAR AND BAR.                        ;[E121]
                                ;       ENTER AT U.RCSR WITH UBAR=xxxLEN TO READ HI CSR,                        ;[E121]
                                ;       ENTER AT U.RCSB WITH UBAR=xxxCOF TO READ LO CSR.                        ;[E121]
                                ;
                                ; INPUTS:
                                ;       UBAR            ADDRESS OF CSR SAVE AREA
                                ;       R2              0 FOR RESPONSE RING OR
                                ;                       NUMBER OF BYTES IN RESPONSE RING FOR COMMAND RING
                                ;
                                ; OUTPUTS:
                                ;       R7              CSR WORD WITH 'OWN' AND 'FLAG' BITS
                                ;       CONDITION CODES SET BY R7 (OWN IS MSB)
                                ;
                                ;       NOTE: READING THE CSR ONE WORD AT A TIME IS DUMB, BUT IT IS THE         ;[E121]
                                ;       ONLY WAY TO PREVENT RACE CONDITIONS IF THE CSR IS NOT LONGWORD           ;[E121]
                                ;       ALIGNED. TRUST ME.                                                      ;[E121]
                                ;;-

                                ; ***   CALCULATE OFFSETS,ETC ***

001737  120445  007005  125574  U.RCSR: INC     UBAR\0,UBAR,BAR %CALL   S.LDR1  ; R1=MAX LENGTH VALUE
001740  120445  007005  135573          INC     UBAR\0,UBAR,BAR %CALL   S.LDRO  ; RO=CURRENT RING PTR
001741  033461  010000  051742          MOV\T   RO,R1           %TNZRO          ; IF RO NEQ 0 THEN R1=CURRENT PTR
001742  013440  007005  131753          MOV     UBAR,BAR        %CALL   U.STOF  ; CALCULATE & SAVE CURRENT OFFSET
001743                                  ASSUME  CMDPOF&17,EQ,<CMDCOF&17>+1      ; MAKE SURE AUTO INCR WILL WORK
001743                                  ASSUME  RSPPOF&17,EQ,<RSPCOF&17>+1      ; MAKE SURE AUTO INCR WILL WORK
001743  031440  000001  131753          DEC     R1,RO           %CALL   U.STOF  ; CALCULATE & SAVE PREVIOUS OFFSET
001744  120445  007005  135573          INC     UBAR\0,UBAR,BAR %CALL   S.LDRO  ; RO=CURRENT OFFSET
001745  030540  000002  101747  U.RCSA: ADD     #2,RO           %JMP    U.RCSC  ; [16K]ADD 2 (BYTES) TO GET HI CSR

                                ; ***   UBAR POINTS AT RSP/CMD-COF ***

001746  013440  007005  125573  U.RCSB: MOV     UBAR,BAR        %CALL   S.LDRO  ; [16K]RO=CURRENT OFFSET

                                ; ***   UBAR POINTS AT RSP/CMD-POF, ADD 2 FOR C/R-CSRW1, RO=OFFSET ***

001747  030545  000002  131777  U.RCSC: ADD     #2,UBAR         %CALL   U.GTAD  ; [16K]GO CALCULATE CSR ENTRY ADDRESS
001750  033740  000001  125635          MOV     #1,RO           %CALL   UNB.RD  ; [16K]READ CSR WORD INTO TEMP STORAGE
001751  111565  017001  145601          SUB\T   #1,UBAR\N,BAR   %JZRO   S.LDR7  ; [16K]R7=CSR WORD
001752  033747  000006  102071          MOV     #ER.RRD,R7      %JMP    U.UERR  ; [16K]IF ERROR THEN FATAL

                                ; ***   ROUTINE TO CALCULATE AND STORE OFFSET ***

001753  070140  000000  000000  U.STOF: ADD\L   RO,RO                           ; RO=4*RSPPTR (CURRENT)
001754  030140  000002  105616          ADD     R2,RO           %JMP    S.STRO  ; ADD IN EXTRA OFFSET/STORE/RETURN
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.UCSR  (UPDATE HOST CSR'S)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       DEPOSITS THE UPDATED HOST CSR BACK IN THE RING BUFFER AND
                                ; DETERMINES IF AN INTERRUPT IS REQUIRED.  USED FOR BOTH THE
                                ; COMMAND AND RESPONSE RINGS.
                                ;
                                ; INPUTS:
                                ;       UBAR --> RING CONTROL AREA (CCSRW1 OR RCSRW1)
                                ;
                                ; OUTPUTS:
                                ;       CONDITION CODES ARE NEGATIVE (OWN=BIT15) IF INTERRUPT REQUIRED
                                ;       HOST CSR'S UPDATED
                                ;
                                ;;-

                                ; ***   FIRST UPDATE RING POINTER ***

001755  111545  007005  125574  U.UCSR: SUB     #<RCSRW1-RSPLEN>,UBAR\N,BAR %CALL S.LDR1; [16K]R1=RING LENGTH
001756  111545  007004  125573          SUB     #<RCSRW1-RSPPTR>,UBAR\N,BAR %CALL S.LDRO; [16K]RO=RING POINTER
001757  011141  000000  000000          SUBC    RO,R1\N                         ; SUB PTR VALUE FROM LENGTH
001760  037760  010000  151761          COM\T   #0,RO                   %TZRO   ; IF EQ THEN RO=-1
001761  130440  000000  135616          INC     RO                      %CALL   S.STRO  ; RO=RO+1/RESET POINTER
001762  111545  007003  135573          SUB     #RCSRW1-RSPCOF,UBAR\N,BAR %CALL S.LDRO  ; [16K]RO=CURRENT CSR ENTRY OFFSET
001763  030540  000002  010000          ADD     #2,RO                           ; [16K]ADJUST RO TO POINT AT HI CSR
001764  013440  007005  125601          MOV     UBAR,BAR                %CALL   S.LDR7  ; [16K]R7=CSR WORD 1
001765  135547  000200  010000          BIC     #OWN,R7                         ; CLEAR OWN BIT
001766  113547  003100  121777          BIS     #FLAG,R7\N,BUF          %CALL   U.GTAD  ; SET FLAG/CALCULATE CSR ENTRY ADDR
001767  033740  000001  136242          MOV     #1,RO                   %CALL   UNB.WR  ; [16K]WRITE UPDATED CSR
001770  033767  010007  052071          MOV\T   #ER.RWR,R7              %JNZRO  U.UERR  ; IF ERROR THE FATAL
001771  114547  000100  010000          BIT     #FLAG,R7                        ; IF HOST DOES NOT WANT INTERRUPTS
001772  111545  017006  167777          SUB\F   #<RCSRW1+1-RSPLEN>,UBAR\N,BAR %RZRO     ; [ECO#2]  THEN EXIT
001773  033740  000001  135610          MOV     #1,RO                   %CALL   S.XORO  ; IF RING LENGTH EQ 1
001774  133740  010200  127777          MOV     #OWN,RO                 %RZRO           ;   THEN RETURN AND FORCE INTERRUPT
001775  131545  007003  010000          SUB     #<RCSRW1+1-RSPPOF>,UBAR,BAR             ; [ECO#2]BAR=PTR TO PREVIOUS RING ENTRY OFFSET
001776  033700  000003  101745          MOV     (BUF),RO                %JMP    U.RCSA  ; RO=PREVIOUS RING ENTRY OFFSET
                                                                                        ; GO CHECK IT AND RETURN OWN BIT
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GTAD (GET CSR ADDRESS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL CALCULATE THE CSR ADDRESS GIVEN THE OFFSET OF
                                ; THE CSR ENTRY IN R0.
                                ;
                                ; INPUTS:
                                ;       R0              OFFSET OF DESIRED CSR ENTRY
                                ;
                                ; OUTPUTS:
                                ;
                                ;
                                ;;-

001777  013740  007162  135574  U.GTAD: MOV     #RNGBSH,BAR     %CALL   S.LDR1  ; R1=ADDR BITS 16 AND 17
002000  013740  007160  125600          MOV     #RNGBSL,BAR     %CALL   S.LDIU  ; IUAR=LO ORDER ADDRESS OF RING
002001  130441  000001  000000          INC     R1                              ; INCR R1
002002  030146  000000  115540          ADD     R0,IUAR         %JMP    S.AXAB  ; SET EXT ADDR BITS/RETURN
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SADQ  (SUBTRACT Q FROM ADDRESS IN IUAR AND R1(RESET CR))
                                ;       U.SADX  (SUBTRACT Q FROM ADDRESS IN IUAR AND R1(RESET CR), MOV R7 TO R1)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL SUBTRACT THE VALUE IN THE Q REGISTER FROM
                                ; THE UNIBUS ADDRESS REGISTER IMAGE (IUAR) AND ADJUST THE EXTENDED
                                ; ADDRESS BITS IN R1 AS REQUIRED.
                                ;
                                ; INPUTS:
                                ;       Q       VALUE TO SUBTRACT FROM IUAR
                                ;       BAR     POINTER TO EXTENDED ADDRESS BITS
                                ;
                                ; OUTPUTS:
                                ;       IUAR,UAR = IUAR - Q
                                ;       POSSIBLY R1 AND CR UPDATED FOR EXTENDED ADDRESS BIT CHANGE
                                ;
                                ;;-

002003  033701  000003  010000  U.SADX: MOV     (BUF),R1                        ; [16K]R1=EXTENDED ADDR BITS
002004  132046  000006  105540  U.SADQ: RSUB    Q,IUAR          %JMP    S.AXAB  ; IUAR,UAR=IUAR-Q
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GPKT (GET MSCP PACKET FROM FREE POOL)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL ATTEMPT TO OBTAIN A MSCP PACKET FROM
                                ; THE MSCP PACKET POOL. IF SUCCESSFUL A ZERO CONDITION CODE WILL BE
                                ; RETURNED AND R10 WILL CONTAIN THE ADDRESS OF THE PACKET QUE WORD
                                ; I.E. THE STARTING PACKET ADDRESS MINUS 1. A NONZERO CONDITION CODE
                                ; WILL BE RETURNED ON A FAILURE TO OBTAIN A BUFFER.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       R10             ADDRESS OF PACKET (IF SUCCESSFUL)
                                ;       CONDITION CODE ZERO IF SUCCESSFUL
                                ;       CONDITION CODE NONZERO IF UNSUCCESSFUL
                                ;
                                ;;-

002005                          U.GPKT: ASSUME  PKTEND&1,EQ,0                   ; MAKE SURE PKTEND IS EVEN
002005  033750  000352  010000          MOV     #PKTEND&LOBYT,R10               ; [16K]CONTRUCT PKTEND
002006  133550  000002  112011          BIS     #PKTEND&HIBYT,R10 %JMP  U.GPKB  ; [16K]R10=END ADDRESS OF POOL
002007  016550  010342  167777  U.GPKA: XOR\F   #PKTBUF,R10\N    %RZRO         ;   THEN RETURN/ELSE CHK FOR LAST PKT
002010  033770  017304  155573          MOV\T   #LOGPKT,R10,BAR %JZRO   S.LDRO  ; IF AT LAST PKT THEN RETURN VALUE OF LOG PKT
002011  131550  007032  125573  U.GPKB: SUB     #MCP.LN+P,R10,BAR %CALL S.LDRO  ; R0=PACKET QUE WORD
002012  114540  000200  112007          BIT     #PSTAT,R0       %JMP    U.GPKA  ; IF PKT STATUS EQ FREE(0)
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.INTR (INTERRUPT HOST)
                                ;       U.INTI (WRITE A 1 AT <RING BASE ADDRESS - Q>)
                                ;       U.VAXP (VAX PURGE CHECK AND INTERRUPT ROUTINE)
                                ;       U.INTV (WRITE R1 AT <RING BASE ADDRESS - Q>)
                                ;       U.INTA DIAGNOSTIC ENTRY POINT
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL GENERATE AN INTERRUPT IF THE VALUE IN 'INTVEC'
                                ; IS NONZERO. THE VALUE IN R0 WILL BE ADDED IN TO THE VALUE IN 'INTVEC'
                                ; TO GIVE A FINAL ADDRESS. IF 'INTVEC' IS ZERO THEN NO INTERRUPTS WILL
                                ; BE GENERATED.
                                ;       THE U.VAXP ROUTINE WILL CHECK THE VAX PURGE BIT AND IF SET
                                ; WILL GENERATE AN INTERRUPT TO INTVEC.
                                ;
                                ; ***   NOTE THAT IF INTERRUPTS ARE DESIRED BY THE HOST, THIS ROUTINE
                                ; ***   WILL WAIT FOREVER FOR UNIBUS INTERRUPT MASTERSHIP. ***
                                ;
                                ; INPUTS:
                                ;       Q       NEGATIVE OFFSET FROM RESPONSE RING BASE (TO WRITE Q)
                                ;       R1      VALUE TO WRITE AT ABOVE ADDRESS (TO INDICATE WHAT KIND OF INTERRUPT)
                                ;       R7      POINTER TO BUFFER FROM WHICH UBAR CAN BE RESTORED (U.VAXP)
                                ;
                                ; OUTPUTS:
                                ;       WORD IN R1 WRITTEN TO RING BASE ADDRESS-Q (FOR U.INTI,U.INTV)
                                ;       INTERRUPT GENERATED TO THE HOST (IF 'INTVEC' NONZERO)
                                ;       Q= NON ZERO - NO ERROR          Q= 0    ERROR, DID NOT RELEASE UNIBUS
                                ; ***   ZRO = 0       NO ERROR          ZRO # 0 ERROR, DID NOT RELEASE UNIBUS
                                ;       UBAR    SDI CONTROL BLOCK POINTER (U.VAXP - IF INTERRUPT GENERATED)
                                ;
                                ;;-

                                ; ***   ENTRY FOR COMMAND AND RESPONSE INTERRUPTS - MAKE R1 = 1 ***
                                ; ***   NEW NPR/INTR TRANSFER ***

002013  033741  000001  102015  U.INTI: MOV     #1,R1           %JMP    U.INTV  ; R1=1,GO WRITE WORD

                                ; ***   ENTRY FOR VAX PURGE CHECK ***

002014  013440  000000  127777  U.VAXP: NOP                     %RTN            ; NO VAX PURGE FOR QDA/BDA [mjt02]

                                ; ***   WRITE VALUE IN R1 TO <RING.BASE ADDRESS - Q> ***

002015  033745  007035  135620  U.INTV: MOV     #ALOLMT,UBAR,BAR %CALL  S.STR1  ; STORE VALUE TO WRITE
002016  034440  000000  121777          CLR     R0              %CALL   U.GTAD  ; GET RING BASE ADDR IN IUAR AND R1
002017  033740  000001  122004          MOV     #1,R0           %CALL   U.SADQ  ; GO SUBTRACT Q FROM RING BASE ADDRESS IN IUAR
002020  013440  000000  136242          NOP                     %CALL   UNB.WR  ; GO WRITE R1 TO AREA PRECEEDING RINGS
002021  033767  010017  042071          MOV\T   #ER.IWR,R7      %JNZRO  U.UERR  ; IF ERROR THEN FATAL
002022  013740  007034  000000          MOV     #FAILUR,BAR                     ; BAR=LAST FAILURE ADDRESS
002023  033702  000014  010000          MOV     (UCRS),R2                       ; [BDAmjt06] SET BIT FOR biic4 goof
002024  035542  000377  010000          BIC     #LOBYT,R2                       ; [mjt06] Low byte MUST be cleared
002025  133542  004001  000000          BIS     #VAXINT,R2,UCRD                 ; [mjt06] SET BIT FOR NONDIAGNOSTIC INTERRUPT
```

```
002026  013740  003010  112030          MOV     #ER.INT,BUF     %JMP    U.INTR  ; SAVE INTERRUPT FAILURE
002027  013440  000000  127777  U.VAXR: NOP                     %RET            ; [BDA] FOR SOMETHING IN OLD UDA CODE

                          U.INTA:
                          U.INTR:
002030  033740  000100  130313  U.INTX: MOV     #BIUCSR,R0      %CALL   RDCMD   ; [BDA] READ INTERRUPT REG
002031  013441  000001  000000          TST     R1                              ; [BDA] IS THE VECTOR = 0?
002032  033742  010002  177777          MOV\F   #BRLV5,R2       %RZRO           ; [BDA] RETURN IF ZERO, ELSE
                                                                                ; [BDA] LOAD BR LEVEL 5 FOR INTER
002033  033740  000100  132050          MOV     #BIUCSR,R0      %CALL   XWRCMD  ; [BDA] GO DO INTERRUPT
                                ;
                                ; MERR LOOKS AT THE EV CODE FOR INTERRUPT COMPLETE.  THERE
                                ; IS NO ERR WHEN MERR IS SET IN THIS CASE
                                ;
002034  013440  110000  006237  1$:     NOP                     %JSTOP  BI.STP  ; [mjt06] IF STOP, EXIT
002035  013740  146220  142034          MOV\F   #<B.NRTY+B.LED>,BREG %JNMERR 1$ ;WAIT FOR INTERRUPT, ELSE
                                                                                ;  LOAD BREG WITH ABORT CMD
002036  013740  006221  010000          MOV     #<B.NRTY+B.LED+B.NABO>,BREG     ;LOAD BREG WITH NO ABORT CMD (to clear merr)

002037  033740  000100  130313          MOV     #BIUCSR,R0      %CALL   RDCMD   ; [BDA] Get current interrupt value
                                ;
                                ; [BDA] BDA NOTE.  IT WILL WAIT FOREVER FOR THE INTERRUPT TO COMPLETE.
                                ;
002040  133742  000002  000000  6$:     MOV     #BRLV5R,R2                      ; [BDA] ONLY HAVE BRLV5R IN R2
002041  033740  000100  132050          MOV     #BIUCSR,R0      %CALL   XWRCMD  ; [BDA] WRITE BRLV5R TO CLEAR INTR /
                                                                ;JMP            ; [BDA] USE RETURN IN WRITE ROUTINE
002042  033702  000014  010000          MOV     (UCRS),R2                       ; [mjt06] GET CONTROL REG AND SEE IF THIS IS A DIAGNOSTIC
002043  114542  000001  010000          BIT     #VAXINT,R2                      ; [mjt06] TEST THE BIT
002044  035542  010377  177777          BIC\F   #LOBYT,R2       %RZRO           ; [mjt06] IF NOT, EXIT/CLEAR LOBYTE
002045  114544  000100  010000          BIT     #DMODE,RLL                      ; [mjt06] if in dm mode NO POLL
002046  135542  014001  027777          BIC     #VAXINT,R2,UCRD %RNZRO          ; [mjt06] AND THEN CLEAR THE BIT AND STORE IT AWAY
002047  033740  000362  100313          MOV     #IPREG,R0       %JMP    RDCMD   ; [mjt06] AND RESET POLL

                                ;
                                ;       The following code was taken out of the interrupt code when
                                ;       Jim found a way of checking the interrupt complete via a test condition
                                ;       This was done so we could get around the pass 4 biic problem with
                                ;       interrupts and no acks
                                ;
                                ;;
                                ;;      XNOR    R7,R7                           ; [BDA] R7 IS TIMEOUT
                                ;;
                                ;;      BIT     #BRLV5S,R2                      ; [BDA]
                                ;;      BIT     #BRLV5R,R2      %JNZRO  5$
                                ;;      MOV     #1000,R2        %JNZRO  6$
                                ;;2$:   DEC     R2              %JNZRO  2$      ; [BDA] WASTE SOME TIME BEFORE ACCESSING THE BI BUS
                                ;;      DEC     R7
                                ;;      MOV\F   #ER.INT,R7      %JNZRO  1$      ; [BDA] ELSE GO TRY AGAIN
                                ;;      NOP                     %JMP    U.UERR  ; [BDA] TIMED OUT/HANG
                                ;;3$:   MOV     #1000,R2                        ; [BDA] WASTE SOME TIME BEFORE GRABBING THE BI BUS
                                ;;4$:   DEC     R2,R2           %JNZRO  4$      ; [BDA]
                                ;;      MOV     #BIUCSR,R0      %CALL   RDCMD   ; [BDA] CHECK TO SEE IF INTERRUPT COMPLETED
```



```
                                ;;5$:   BIT     #BRLV5R,R2                      ; [BDA]
                                ;;      NOP                     %JZRO   3$      ; [BDA] IF NOT SET, TRY AGAIN
                                ;;      MOV     #BIUCSR,R0      %CALL   RDCMD   ; [BDA] SEE IF INTERRUPT WAS SENT
```

```
                                ;;*****************************************************************************
                                ;
                                ;       SUBROUTINE xWRCMD
                                ;
                                ;       xWRCMD  will issue a write command in NORMAL mode
                                ;
                                ;       Input:
                                ;               R0 contains BI WRITE command literal.
                                ;               R1 contains data to be loaded into the LO word of a BI REGISTER.
                                ;               R2 contains data to be loaded into the HI word of a BI REGISTER.
                                ;       Output:
                                ;               RAM location RGDATL will get data from R1.
                                ;               RAM location RGDATH will get data from R2.
                                ;               BREG will have LOOPBACK enabled if this routine was entered
                                ;                   at location WRLCMD.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;
                                ;;*****************************************************************************

002050  013440  000000  130265  XWRCMD: NOP                     %CALL   BRGADR  ;GO LOAD BIIC REGISTER ADDRESS
002051  033740  000004  000000          MOV     #BIWRCM,R0                      ;GET WRITE COMMAND
002052  033440  005540  000000          MOV     R0,,BCAID   @LCOM               ;LOAD COMMAND
002053  013740  007271  000000          MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
002054  033441  003001  135621          MOV     R1,,BUF         %CALL   S.STR2  ;STORE REGISTER DATA
002055  013740  007271  000000          MOV     #RGDATL,\N,BAR                  ;POINT TO LO REG DATA LOC IN RAM
002056  013700  005523  000000          MOV     (BUF),BCAID @WRFST              ;WRITE FIRST BUFFER
002057  013740  117272  006237          MOV     #RGDATH,\N,BAR  %JSTOP  BI.STP  ;JUMP IF BI STOPPED /                   [mr1001]
                                                                                ; POINT TO HI REG DATA LOC IN RAM
002060  013700  005743  010000          MOV     (BUF),BCAID @LBWR               ;WRITE LAST BUFFER AND DO THE WRITE /
                                                                                ; GO WAIT FOR CMD COMPLETE & ERROR FREE /
                                                                                ; RETURN
```



```
                                ;;*****************************************************************************
                                ;
                                ;       WAIT FOR COMMAND COMPLETE AND ERROR FREE
                                ;
                                ;       This routine will  wait approximately 10. seconds  before flagging a BI
                                ;       COMMAND TIMEOUT error.
                                ;
                                ;       Upon entering this routine,  MERR is checked  before  issuing a BI READ
                                ;       or WRITE COMMAND.   If MERR is set,  the command  is never started.  If
                                ;       MERR is clear,  wait for NCDONE (Command Not Done), which indicates the
                                ;       BI COMMAND has started.  If this does not occur within a certain amount
                                ;       of time,  then retry the command once.  If NCDONE is not true after one
                                ;       retry,  then  a BI CMD TIMEOUT error is reported.  When NCDONE is true,
                                ;       then wait for  CDONE (Command Done)  to be asserted.   If  CDONE is not
                                ;       detected within 10.sec,  a BI CMD TIMEOUT error  is reported.  If CDONE
                                ;       is detected,  the  routine  checks MERR (Master Error) and exits.
                                ;
                                ;       Input:
                                ;               R0 contains the BI COMMAND literal (BIRDCM or BIWRCM).
                                ;       Output:
                                ;               R1 & R2 will be destroyed.
                                ;
                                ; NOTE: Errors in this routine display the following in the SA register;
                                ;       106340 - BI COMMAND TIMOUT error
                                ;
                                ;;*****************************************************************************

002061  013440  140000  010037  XWTCMD: NOP                     %JMERR  ERRB1M  ;ERROR IF MASTER ERROR /                [mr1002]
                                                                                ; SETUP TIMEOUT COUNT TO BE SHORT
                                        ;Wait for BI COMMAND DONE
002062  013440  160000  002065          NOP                     %JCDONE 6$      ;JUMP IF CMD DONE /                     [mr1002]
                                                                                ; SETUP CMD CMPL OUTER TIMEOUT COUNT
002063  013440  160000  112063  5$:     NOP                     %JNCDONE 5$     ;JUMP IF CMD DONE /                     [mr1002]
                                                                                ; DONE WITH CMD CMPL INNER WAIT LOOP YET?
002064  013440  140000  010037          NOP                     %JMERR  ERRB1M  ; IF ERROR/AFTER COMMAND DONE/BRANCH
002065  013440  000000  127777  6$:     NOP                     %RET
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.UERR (BUS CONTROLLER INTERNAL ERROR HANDLER)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SAVE THE ERROR CODE PASSED IN R7 IN
                                ; THE CONTROLLER RAM LOCATION DEFINED BY 'FAILUR', IN ADDITION IT
                                ; WILL ATTEMPT TO ENCODE THE UPPER BYTE IN AN UNIQUE MANNER.
                                ; IT WILL THEN SET THE ERROR CODE IN SA AND SET BIT 15 OF SA
                                ; (TO INDICATE AN ERROR) AND HANG AWAITING A REINITIALIZATION.
                                ;
                                ; INPUTS:
                                ;       R7              CONTROLLER FATAL ERROR CODE
                                ;
                                ; OUTPUTS:
                                ;       SA REG          CONTAINS ERROR CODE AND BIT 15 SET
                                ;       FAILUR          CONTAINS ERROR CODE AND UNIQUE IDENTIFIER
                                ;
                                ;;-

                                ; ***   ENTRY POINT FOR D.PROC ERROR - R11 HAS ERROR CODE ***

002066  033447  000011  102071  U.UERD: MOV     R11,UER         %JMP    U.UERR  ; PUT ERROR CODE IN UER
002067  114544  000002  010000  U.HERR: BIT     #INDIAG,RLL                     ; [16K]IF IN DIAGNOSTICS
002070  013460  010620  077777          NOP\T           @RCLR   %RNZRO          ; [U52EC2][16K]  THEN RETURN TO THEM
002071  013740  007034  135625  U.UERR: MOV     #FAILUR,BAR     %CALL   S.STR7  ; SAVE CODE IN FAILUR
002072  013740  006310  010000          MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG     ;[BDA]LOAD BREG WITH ABORT COMMAND
002073  133547  000200  010000          BIS     #ERRBIT,UER                     ;[BDA]WRITE SA WITH ERR BIT & CODE
002074  033441  000007  130366          MOV     UER,R1          %CALL   WRTERR  ;[BDA]WRITE ERROR TO SA REGISTER
002075  013740  006310  010000          MOV     #<B.NRTY+B.LOOP+B.BAD>,BREG     ;[BDA]LOAD BREG WITH ABORT COMMAND
002076  013447  000007  112076          TST     UER             %JMP    .       ;[BDA]HANG FOREVER TESTING ERROR REG
```



```
                                .SBTTL  SEEK ORDERING ROUTINE

                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SEKO (SEEK ORDERING ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS SEEK ORDERING USING AN ELEVATOR
                                ; ALGORITHM THAT HAS THE FOLLOWING PROPERTIES:
                                ;       1. ONLY NON-SEQUENTIAL MSCP COMMANDS ARE ORDERED.
                                ;          SEQUENTIAL COMMANDS WILL REMAIN IN THE SAME RELATIVE
                                ;          POSITION AS WHEN FIRST QUEUED.
                                ;       2. IF THE TOP OF THE LIST IS A SEQUENTIAL COMMAND THEN
                                ;          IT WILL BE CHOSEN.
                                ;       3. WITHIN THE LIST OF NON-SEQUENTIAL COMMANDS (I.E. THE SET
                                ;          OF NON-SEQUENTIAL COMMANDS IS BOUNDED BY THE FIRST SEQUENTIAL
                                ;          COMMAND ENCOUNTERED) THE SELECTION CRITERIA ARE AS FOLLOWS:
                                ;               A. AN EXPRESS COMMAND IS THE HIGHEST PRIORITY.
                                ;               B. A COMMAND ON THE CURRENT CYLINDER IS NEXT.
                                ;               C. A COMMAND WHICH IS CLOSEST TO THE CURRENT CYLINDER
                                ;                  IN THE CURRENT SEEK DIRECTION IS NEXT (DIRECTION
                                ;                  STAYS THE SAME).
                                ;               D. A COMMAND WHICH IS CLOSEST TO THE CURRENT CYLINDER
                                ;                  IN THE OPPOSITE SEEK DIRECTION IS NEXT (DIRECTION
                                ;                  IS REVERSED).
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO ACTIVE SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       POTENTIALLY REORDERED MSCP PACKET QUEUE
                                ;       SDI.PQ CONTAINS THE CHOSEN COMMAND
                                ;       Q,R0,R1,R2,R3,R6,R7,R10 ARE USED AS TEMPORARY REGISTERS
                                ;
                                ;;-

002077  010545  017026  002112  U.SEKA: ADD     #SDI.ES,UBAR\N,BAR %JNZRO U.SEKC; IF MORE PKTS THEN LOOP
002100  136542  063200  152150          XOR\F   #ELEV,R2,BUF    %JUPF   U.SEKH  ; IF UPF SET THEN DONE ELSE REVERSE DIRECTION

                                ; ***   SEEK ORDERING ENTRY POINT - INITIALIZE SEEK ORDERING VARIABLES ***

002101  000545  007022  125602  U.SEKO: ADD     #SDI.PQ,UBAR,Q,BAR %CALL S.LD10 ; R10=PKT QUE HEAD,Q=PTR TO HEAD
002102  133747  010200  137777          MOV     #BIT15,R7       %RZRO           ; IF ZERO THEN NO PKTS/RETURN
                                                                                ; INIT BEST CYL DIFF TO MEDIAN NUMBER

                                ; ***   CHECK AND UPDATE FAIRNESS COUNTER ***

002103  010545  007026  125573  U.SEKI: ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDRO ; [U52EC1]R0=FAIRNESS COUNTER
002104  014540  000002  010000          BIT     #OFFTRK,R0                      ; IF NO ERROR RECOVERY LEVELS DONE PREVIOUS COMMAND
002105  035540  010002  152107          BIC\F   #OFFTRK,R0      %JZRO   U.SEKB  ;   THEN CONTINUE
002106  033540  003004  102103          BIS     #FSEEK,R0,BUF   %JMP    U.SEKI  ;   [U52EC1]ELSE SET FORCE SEEK FLAG
002107  114540  000020  000000  U.SEKB: BIT     #BIT12,R0                       ; IF LIMIT EXCEEDED
002110  130540  010377  052150          ADDC\F  #377,R0         %JNZRO  U.SEKH  ;   THEN TAKE TOP OF QUE
002111  033243  000603  125616          MOV     Q,R3    @RUPF   %CALL   S.STRO  ;   ELSE RESET COUNTER/INIT PREV. PTR,ZAP WIN FLAG
```

```
                                ; ***   IN THE FOLLOWING CODE, R3 HAS POINTER TO PREVIOUS LIST ELEMENT,
                                ; ***   Q HAS POINTER TO ELEMENT BEFORE 'BEST' ELEMENT, UPF SET IF Q IS VALID
                                ; ***   CHECK FOR SEQUENTIAL COMMAND ***

002112  010550  007006  125573  U.SEKC: ADD     #P.OPCD,R10\N,BAR %CALL S.LDR0  ; R0=OP CODE                                    ;[E121]
002113  010550  007007  125574          ADD     #P.MOD,R10\N,BAR %CALL  S.LDR1  ; R1=MSCP MODIFIERS
002114                                  ASSUME  MD.EXP,EQ,BIT15                 ; MAKE SURE MD.EXP IS MSB
002114  014540  030010  102147          BIT     #SEQUEN,R0      %JMSB   U.SEKG  ; IF EXPRESS THEN DONE                          ;[E121]
                                                                                ; IF SEQUENTIAL
002115  013440  010000  012150          NOP                     %JNZR0  U.SEKH  ;   THEN DONE                                  ;[E121]
002116  010545  007025  135574          ADD     #SDI.OM,UBAR\N,BAR %CALL  S.LDR1; if overlapped operation in progress          ;[E121]
002117  014540  010040  112122          BIT     #<OP.RD&OP.WR&OP.CMP>,R0 %JZR0 5$;    then only allow read, write, or cmp;[E121]
002120  010545  017026  052122          ADD\F   #SDI.ES,UBAR\N,BAR %JNZR0 5$    ;    ELSE discard this candidate               ;[E121]
002121  033702  000003  112145          MOV     (BUF),R2        %JMP 30$        ;         and set up R2 as expected           ;[E121]
                                5$:                                             ;                                             ;[E121]
002122  010550  007024  125575          ADD     #S.CYLL,R10\N,BAR  %CALL S.LDR2 ; R2=PACKET LO CYL                              ;[E121]
002123  010545  007011  135573          ADD     #SDI.CL,UBAR\N,BAR %CALL S.LDR0 ; R0=CURRENT LO CYL
002124  010545  007012  125574          ADD     #SDI.CH,UBAR\N,BAR %CALL S.LDR1 ; R1=CURRENT HI CYL
002125  132140  000002  000000          RSUB    R2,R0                           ; R0=CURRENT & PKT LO DIFF
002126  130461  020001  052127          INC\T   R1              %TNCRY          ; PROPAGATE BORROW
002127  010550  007025  135575          ADD     #S.CYLH,R10\N,BAR  %CALL S.LDR2 ; R2=PACKET HI CYL
002130  132141  000002  010000          RSUB    R2,R1                           ; DIFFERENCE NOW IN (R1,R0)
002131  010545  007026  125575          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDR2 ; R2=ELEVATOR DIRECTION
002132                                  ASSUME  ELEV,EQ,BIT15                   ; MAKE SURE ELEVATOR BIT IS MSB
                                                ;the following code [rae2-c119] was rewritten to make           [rae2-c119]
                                                ;it clearer and to update the best difference only              [rae2-c119]
                                                ;when a better difference was found, previously it was          [rae2-c119]
                                                ;updated on a <= causing the most recent packet to be           [rae2-c119]
                                                ;selected rather than the oldest when packets were for          [rae2-c119]
                                                ;the same cylinder.                                             [rae2-c119]
002132  037441  030001  052136          COM\F   R1              %JNMSB 10$      ; if elevator going down negate diff    [rae2-c119]
002133  037440  000000  000000          COM     R0                              ;                                       [rae2-c119]
002134  130440  000000  000000          INC     R0                              ;                                       [rae2-c119]
002135  130461  020001  142136          INC\T   R1              %TCRY           ; (ripple the carry if no low bits)     [rae2-c119]
002136  112141  000007  000000  10$:    CMP     R7,R1                           ;                                       [rae2-c119]
002137  013480  020000  042145          NOP\T                   %JLESS 30$      ; if BEST(hi) < CUR(hi) then get next   [rae2-c119]
002140  112146  010000  042142          CMP\F   R0,R6           %JNZR0 20$      ; if BEST(hi) > CUR(hi) then update BEST[rae2-c119]
                                                                                ; if BEST(hi) = CUR(hi) cmp  lo's       [rae2-c119]
002141  013440  020000  112145          NOP                     %JCRY  30$      ; if CUR(lo) >= BEST(lo) then get next  [rae2-c119]
                                20$:                                            ; else update BEST diff                 [rae2-c119]
002142  033447  000001  010000          MOV     R1,R7                           ; SO SET BEST DIFF=NEW DIFF             [rae2-c119]
002143  033446  000000  010000          MOV     R0,R6                           ;                                       [rae2-c119]
002144  003443  000403  000000          MOV     R3,Q    @SUPF                   ; SET BEST PTR AND FLAG                 [rae2-c119]
                                30$:                                            ;                                       [rae2-c119]
002145  033443  007010  125602          MOV     R10,R3,BAR      %CALL   S.LD10  ; CHAIN TO NEXT, REMEMBERING PREVIOUS   [rae2-c119]
002146  115550  000300  112077          BIC     #Q.STAT,R10\N   %JMP    U.SEKA  ; TEST FOR END OF LIST AND LOOP

                                ; ***   EXPRESS REQUEST EXIT POINT ***

002147  003443  000003  010000  U.SEKG: MOV     R3,Q                            ; EXPRESS COMMANDS ALWAYS WIN

                                ; ***   NORMAL EXIT POINT - MOVE WINNER TO HEAD OF LIST ***
```



```
002150  013240  007600  125602  U.SEKH: MOV     Q,BAR   @RUPF    %CALL   S.LD10  ; R10=WINNER
002151  013440  007010  135575          MOV     R10,BAR         %CALL   S.LDR2  ; R2=SUCCESSOR OF WINNER
002152  013240  007000  135621          MOV     Q,BAR           %CALL   S.STR2  ; UNLINK WINNER FROM QUE
002153  013440  000000  133230          NOP                     %CALL   U.LNKH  ; [U52EC1]LINK R10 TO HEAD OF QUEUE

                                ; ***   SET UP ERROR RETRY/RECOVERY CONTROL WORDS ***

002154  010545  007017  125630          ADD     #SDI.E1,UBAR\N,BAR %CALL S.CLRB ; CLR LEVEL 1 ERROR
002155  010545  007020  135630          ADD     #SDI.E0,UBAR\N,BAR %CALL S.CLRB ; CLR LEVEL 0 ERROR
002156  010545  007003  125630          ADD     #SDI.SW,UBAR\N,BAR %CALL S.CLRB ; ZAP STATUS WORD
002157  137746  000035  132375          COM     #<DRAVL!DRVOL!DRDUP!DATT>,R6 %CALL U.CLRS; ZAP MOST BITS IN SDI.ST (JUST IN CASE!)
002160  010545  007025  135600          ADD     #SDI.OM,UBAR\N,BAR %CALL S.LDR6 ; [U52EC1]R6=OVERLAPPED COMMAND
002161  033766  010100  042162          MOV\T   #BFSV,R6        %TNZR0          ; [U52EC1]IF OVERLAPPING THEN SET BUFR SERVICE
002162  033546  000041  123102          BIS     #<PKIP+SUSP>,R6 %CALL   U.SETS  ; [U52EC1]SET PKT IN PROG AND SUSPEND I/O IN SDI.ST

                                ; ***   CHECK FOR COMMAND ABORTED ***

002163  010550  007001  135573          ADD     #P.VCID,R10\N,BAR %CALL S.LDR0  ; [16K]R0=ABORT FLAG WORD
002164                                  ASSUME  PABRT,EQ,BIT00                  ; [16K]MAKE SURE PABRT IS LSB
002164  014546  040100  112166          BIT     #BFSV,R6        %JNLSB  U.OPCD  ; [16K][U52EC1]IF NOT ABORTED THEN CONTINUE
002165  033760  010002  143641          MOV\T   #ST.ABO,R0      %JZR0   U.DONE  ; [U52EC1]IF NOT OVERLAPPING THEN ABORT CMD
002166                                  ASSUME  U.OPCD,EQ,.                     ; [U52EC1]  ELSE IGNORE ABORT & FALL INTO U.OPCD
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.OPCD (OP CODE PROCESSOR ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL ISOLATE THE OP CODE CODE FOR A MSCP PACKET,
                                ; VERIFY IT AND VECTOR TO THE ROUTINE THAT HANDLES THAT OP CODE.
                                ;
                                ; INPUTS:
                                ;       R10             ADDRESS OF FIRST WORD OF MSCP PACKET
                                ;
                                ; OUTPUTS:
                                ;       R0 IS USED AS A TEMPORARY REGISTER
                                ;
                                ;;-

002166  010550  007006  125604  U.OPCD: ADD     #P.OPCD,R10\N,BAR %CALL S.LLBO  ; [U52EC1]R0=OP CODE
002167  013440  007610  125574           MOV     R10,BAR @RUPF   %CALL   S.LDR1  ; R1=MSCP PKT STATUS

                                ; ***   CHECK VIRTUAL CIRCUIT IDENTIFIER ***

002170  010550  007001  135575           ADD     #P.VCID,R10\N,BAR %CALL S.LDR2  ; R2=VIRTUAL CIRCUIT ID
002171                                   ASSUME  <PKTO20&BIT00>,EQ,0             ; [16K]MAKE SURE CHECK IS VALID
002171                                   ASSUME  <<SDI.1!SDI.2!SDI.3!SDI.4>&BIT00>,NE,0; [16K]DITTO
002171                                   ASSUME  <SDI.AT&BIT00>,EQ,0             ; [16K]DITTO

                                ; ***   IF CONTROLLER INTERNAL COMMAND THEN SKIP VC CHECK ***

002171  014550  000001  000000           BIT     #BIT00,R10                      ; [16K]IF CONTROLLER INTERNAL PKT
002172  035542  010017  042336           BIC\F   #LONIB,R2       %JNZRO  U.ATTA  ;   THEN GO PROCESS/ELSE ISOLATE HI 24 BITS OF VC ID
002173                                   ASSUME  VC.CMD,EQ,0                     ; VALIDATE VIR. CIR. ASSUMPTION
002173                                   ASSUME  VC.DMM,EQ,177760                ; VAILDATE VIR. CIR. ASSUMPTION
002173  017542  010017  002224           XNOR    #LONIB,R2\N     %JNZRO  U.OPDM  ; IF NEQ 0 THEN COMPLEMENT AND CHK FOR DM VIRT CIR.

                                ; ***   VIRTUAL CIRCUIT 0 - MSCP COMMANDS ***
                                ; ***   CHECK FOR BETTER BE IMMEDIATE COMMAND ***

002174  016550  000304  010000           XOR     #LOGPKT,R10\N                   ; IF PACKET EQ LOGPKT

                                ; ***   IF VIRTUAL CIRCUIT 0 AND IN DM MODE THEN FATAL ERROR ***

002175  114544  010100  102215           BIT     #DMODE,RLL      %JZRO   U.OPIM  ;   [16K]THEN BETTER BE IMMEDIATE/IF IN DM MODE
002176  033767  010016  052071  U.VCER:  MOV\T   #ER.VCI,R7      %JNZRO  U.UERR  ;   THEN FATAL VIRTUAL CIRCUIT ERROR

                                ; ***   MSCP NON-SEQUENTIAL COMMANDS ***

002177  016540  000041  010000           XOR     #OP.RD,R0\N                     ; CHECK FOR READ
002200  016540  010042  112664           XOR     #OP.WR,R0\N     %JZRO   U.RD    ; CHECK FOR WRITE               [rae02]
002201  016540  010040  112666           XOR     #OP.CMP,R0\N    %JZRO   U.WR    ; CHECK FOR COMPARE             [rae02]
002202  016540  010020  102662           XOR     #OP.ACC,R0\N    %JZRO   U.CMPR  ; CHECK FOR ACCESS              [rae02]

                                ; ***  ACC, ERS, RPL must verify reserved fields, hence call U.IOPO        [rae02]
```



```
002203  016540  010022  112670           XOR     #OP.ERS,R0\N    %JZRO   U.ACC   ; CHECK FOR ERASE                [rae02]
002204  016540  010024  102672           XOR     #OP.RPL,R0\N    %JZRO   U.ERS   ; CHECK FOR REPLACE              [rae02]
002205  013460  010400  142674           NOP\T           @SUPF   %JZRO   U.RPL   ; IF REPLACE THEN SET @SUPF[rae02]
002206  016540  000023  000000           XOR     #OP.FLU,R0\N                    ; CHECK FOR FLUSH
002207  016540  010021  112272           XOR     #OP.CCD,R0\N    %JZRO   U.FLU   ; CHECK FOR COMPARE CONTROLLER DATA

                                ; ***   MSCP SEQUENTIAL COMMANDS ***

002210  016540  010010  112270           XOR     #OP.AVL,R0\N    %JZRO   U.CCD   ; CHECK FOR AVAILABLE
002211  016540  010011  102344           XOR     #OP.ONL,R0\N    %JZRO   U.AVAL  ; CHECK FOR ONLINE
002212  016540  010012  103424           XOR     #OP.SUC,R0\N    %JZRO   U.ONLN  ; CHECK FOR SET UNIT CHAR
002213  016540  010013  113422           XOR     #OP.DAP,R0\N    %JZRO   U.SUCH  ; CHECK FOR ACCESS PATHS
002214  013440  010000  102346           NOP                     %JZRO   U.DAP   ; IF DAP THEN USE AVAIL ROUTINE

                                ; ***   IMMEDIATE COMMANDS ***

002215  016540  000001  000000  U.OPIM:  XOR     #OP.ABO,R0\N                    ; CHECK FOR ABORT
002216  016540  010002  102274           XOR     #OP.GCS,R0\N    %JZRO   U.ABRT  ; CHECK FOR GET COMMAND STATUS
002217  016540  010003  112515           XOR     #OP.GUS,R0\N    %JZRO   U.GCST  ; CHECK FOR GET UNIT STATUS
002220  016540  010004  112617           XOR     #OP.SCC,R0\N    %JZRO   U.GUST  ; CHECK FOR SET CONTROLLER CHAR
002221  016550  010304  113653           XOR     #LOGPKT,R10\N   %JZRO   U.SCCH  ; IF R10 EQ LOGPKT
002222  033747  010012  112071           MOV     #ER.NIM,R7      %JZRO   U.UERR  ;   THEN HARD FAILURE

                                ; ***   ENTRY POINT FOR OP CODE ERROR EXIT ***

002223  013440  000000  102231  U.OPCE:  NOP                     %JMP    U.CMER  ;   ELSE COMMAND ERROR

                                ; ***   VIRTUAL CIRCUIT 177760 - CONTROLLER SPECIFIC MAINTENANCE COMMANDS ***

002224  116542  010002  003721  U.OPDM:  XOR     #DUPVC,R2\N     %JNZRO  U.STUD  ; IF R2 NEQ 177760 THEN CHECK DUP
002225  010550  007020  135575           ADD     #P.RGIO,R10\N,BAR %CALL S.LDR2  ; COMMON ARG FOR MAINT RD, WR
002226  016540  000030  010000           XOR     #OP.MRD,R0\N                    ; CHECK FOR MAINTENANCE READ
002227  016540  010031  113406           XOR     #OP.MWR,R0\N    %JZRO   U.MNRD  ; CHECK FOR MAINTENANCE WRITE
002230  036542  010001  103413           XOR     #1,R2           %JZRO   U.MNWR  ; IF NO MATCH
002231  010550  007006  135630  U.CMER:  ADD     #P.OPCD,R10\N,BAR %CALL S.CLRB  ; [U52EC3]IF ILLEGAL OP CODE THEN ZAP IT
002232  033741  000004  112506           MOV     #<P.OPCD-P>,R1  %JMP    U.CMDA  ; R1=MSCP OFFSET/FINISH UP
```

```
                                ;  ***  [rae02] start of change
                                ;;+
                                ;  ROUTINE NAME:
                                ;       U.RTST:  (reserved field test)
                                ;
                                ;  FUNCTIONAL DESCRIPTION:
                                ;       This routine tests that a series of words are zero.
                                ;
                                ;  INPUTS:
                                ;       R1  = word offset within Packet of first word of field to be verified.
                                ;       R10 = packet address
                                ;         Q = count of sequential words to be verified.
                                ;
                                ;  OUTPUTS:
                                ;       NZRO    if in error
                                ;           R1 = word offset to field in error
                                ;            Q = destroyed
                                ;
                                ;       ZRO     if no error
                                ;           R1 = 0
                                ;            Q = destroyed
                                ;;-
002233  010150  007001  010000  U.RTST: ADD     R1,R10\N,BAR                    ; calc adrs
002234  013700  000003  010000          TST     (BUF)                           ; and test it
002235  001240  010000  077777          DEC\F   Q               %RNZRO          ; if not zero return
002236  130441  010001  012233          INC     R1              %JNZRO  U.RTST  ; if Q not zero then loop
002237  034441  000001  127777          CLR     R1              %RET            ; if done clear R1, set ZRO, and return

002240  013700  000003  137777  S.TST:  TST     (BUF)           %RET            ; test contents of (BUF)
```



```
                                ;;+
                                ;  ROUTINE NAME:
                                ;       U.RSVA:         (test reserved words 6, 16, 17, and P.BUF0-P.BUF5)
                                ;       U.RSVB:         (test reserved words                P.BUF0-P.BUF5)
                                ;
                                ;  FUNCTIONAL DESCRIPTION:
                                ;       These routines will test reserved fields common to several op codes.
                                ;
                                ;  INPUTS:
                                ;       R10    = packet address
                                ;
                                ;  OUTPUTS:
                                ;       ZRO  - if no error
                                ;           R1  = restored to packet status
                                ;            Q  = destroyed
                                ;       NZRO - if error
                                ;           R1  = word offset to field in error.
                                ;            Q  = destroyed
                                ;;-
002241  033741  000010  010000  U.RSVA: MOV     #P.RS06,R1                      ;test reserved field at word 6
002242  010150  007001  122240          ADD     R1,R10\N,BAR    %CALL  S.TST    ;test it/ if not zero return
002243  033741  010006  027777          MOV     #<P.RS06-P>,R1  %RNZRO          ; load offset for status return [C118]
002244  033741  000022  000000          MOV     #P.RS16,R1                      ;test reserved field            [C118][E122]
002245  003740  000002  122233          MOV     #2,Q            %CALL  U.RTST   ;for two words
002246  033741  010020  037777          MOV     #<P.RS16-P>,R1  %RNZRO          ;if failed then return with offset [E122]

002247  033741  000012  000000  U.RSVB: MOV     #P.BUF0,R1                      ;test the reserved fields BUF0
002250  003740  000006  132233          MOV     #6,Q            %CALL  U.RTST   ;thru BUF5
002251  033741  010010  037777          MOV     #<P.BUF0-P>,R1  %RNZRO          ;if failed then return with offset
002252  013440  007010  125574          MOV     R10,BAR         %CALL  S.LDR1   ;restore R1
002253  004240  000000  127777          CLR     Q               %RET            ;set ZRO and return
```

```
                                ;;+
                                ;  ROUTINE NAME:
                                ;       MLN20   test MSCP msg leng for >= 20
                                ;       MLN40   test MSCP msg leng for >= 40
                                ;       MLN44   test MSCP msg leng for >= 44
                                ;
                                ;  FUNCTIONAL DESCRIPTION:
                                ;       These routine test the length of the MSCP message in the packet for
                                ;       at least the valid record length.
                                ;
                                ;  INPUTS:
                                ;       R10     - Packet address
                                ;       MSCPLN  - length of MSCP message
                                ;
                                ;  OUTPUTS:
                                ;       ZRO     -   successful - length is ok
                                ;      NZRO     - unsuccessful - length is bad
                                ;        Q      - destroyed
                                ;;-

002254  003740  000017  112257 MLN20:  MOV     #20-1,Q         %JMP    MLNTST  ;set test value to 20
002255  003740  000037  102257 MLN40:  MOV     #40-1,Q         %JMP    MLNTST  ;set test value to 40
002256  003740  000043  102257 MLN44:  MOV     #44-1,Q         %JMP    MLNTST  ;set test value to 44
002257  013740  007264  010000 MLNTST: MOV     #MSCPLN,BAR                     ;cmp against MSCP msg len
002260  101600  000003  010000         SUB     (BUF),Q                         ;
002261  004240  030000  127777         CLR     Q               %RNEG           ;if leng >= min then return ZRO
002262  100240  000000  127777         INC     Q               %RET            ;if leng <  min then return NZRO
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       TSTMOD  - Test the MSCP command modifiers for validity.
                                ;
                                ; INPUTS:
                                ;       R1  - mask of legal modifiers
                                ;
                                ; OUTPUTS:
                                ;       ZRO  - if reserved modifiers were not set
                                ;       NZRO - if reserved modifiers were set
                                ;       Q    - destroyed
                                ;       R1   - reset to MSCP packet status
                                ;;-

002263  033241  000001  010000 TSTMOD: MOV     Q,R1                            ; (Q can't be R src of ALU)
002264  010550  007007  125572         ADD     #P.MOD,R10\N,BAR %CALL  S.LDQQ  ; get Modifier
002265  005041  000001  000000         BIC     R1,Q                            ; turn off all legal bits
002266  013440  007010  125574         MOV     R10,BAR         %CALL   S.LDR1  ; restore R1
002267  013240  000000  127777         TST     Q               %RET            ; set condition codes and return
                                ;  ***  [rae02]  end  of change
                                .PAGE
```

```
                                .SBTTL  MSCP AND DUP COMMAND ROUTINES

                                ; ROUTINE NAME:
                                ;       U.ABRT  (MSCP ABORT COMMAND)
                                ;       'IMMEDIATE COMMAND'
                                ;       U.NOP   (MSCP FLUSH AND COMPARE CONTROLLER DATA COMMANDS)
                                ;       'NON-SEQUENTIAL COMMANDS TREATED AS IMMEDIATE COMMANDS'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PROCESSES THE MSCP 'ABORT' COMMAND. IT WILL PERFORM
                                ; THE FOLLOWING FUNCTIONS:
                                ;       1. CROSS REFERENCE OUTSTANDING REFERENCE NUMBER TO THE CORRECT PACKET
                                ;       2. IF A REFERENCE NUMBER MATCH IS FOUND IT WILL SET THE 'PABRT'
                                ;          BIT IN THE PACKET STATUS AND LINK WORD.
                                ;
                                ;       THIS ROUTINE PERFORMS A 'NOP' FUNCTION FOR THE FOLLOWING
                                ; MSCP COMMANDS:
                                ;       FLUSH
                                ;       COMPARE CONTROLLER DATA
                                ;
                                ; INPUTS:
                                ;       R1O             POINTER TO MSCP ABORT PACKET
                                ;
                                ; OUTPUTS:
                                ;       IF REFERENCE NUMBER MATCH THEN PACKET IS MARKED FOR ABORTION
                                ;
                                ;-
127400                          M.CCDH  :=      MD.EXP!MD.CSE!MD.SCH!MD.SCL!MD.SEC!MD.SER       ;OP.CCD modifiers    [rae02]
000200                          M.CCDL  :=      MD.SSH                                          ;OP.CCD modifiers    [rae02]
002270  003740  000200  000000  U.CCD:  MOV     #M.CCDL,Q                       ; set legal modifier bits            [rae02]
002271  103640  000257  102304          BIS     #M.CCDH,Q       %JMP    U.NOP   ;                                    [rae02]

121400                          M.FLUH  :=      MD.EXP!MD.CSE!MD.SEC!MD.SER                     ;OP.FLU modifiers    [rae02]
000200                          M.FLUL  :=      MD.SSH                                          ;OP.FLU modifiers    [rae02]
002272  003740  000200  000000  U.FLU:  MOV     #M.FLUL,Q                       ; set legal modifier bits            [rae02]
002273  103640  000243  102304          BIS     #M.FLUH,Q       %JMP    U.NOP   ;                                    [rae02]

000000                          M.ABRT  := 0                                    ;no modifiers allowed for ABORT      [rae02]
                                U.ABRT:
002274  013440  000000  132254          NOP                     %CALL   MLN2O   ; test that msg leng is large enough [rae02]
002275  034441  010001  002506          CLR     R1              %JNZRO  U.CMDA  ;   if not declare error            [rae02]
002276  003740  000000  132263          MOV     #M.ABRT,Q       %CALL   TSTMOD  ; test for reserved modifier bits    [rae02]
002277  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error              [rae02]
002300  013440  000000  132536          NOP                     %CALL   U.GREF  ; GO MATCH REFERENCE NUMBER
002301  033703  010003  142303          MOV\F   (BUF),R3        %JZRO   U.IMEX  ; [16K]IF NO MATCH THEN EXIT
002302  033543  003001  010000          BIS     #PABRT,R3,BUF                   ; [16K]  ELSE ABORT THIS PACKET

                                ; ***   IMMEDIATE AND NOP COMMAND SUCCESS EXIT POINT ***

002303  033740  000000  112420  U.IMEX: MOV     #ST.SUC,R0      %JMP    U.C2EP  ; R0=SUCCESS/CHANGE TO END PKT/EXIT
                                U.NOP:                                          ;                                    [rae02]
002304  013440  000000  122263          NOP                     %CALL   TSTMOD  ; test modifier bits                 [rae02]
002305  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error              [rae02]
002306  013440  000000  122255          NOP                     %CALL   MLN4O   ; test that msg leng is large enough [rae02]
```



```
002307  034441  010001  002506          CLR     R1              %JNZRO  U.CMDA  ;   if not declare error            [rae02]
002310  013440  000000  122247          NOP                     %CALL   U.RSVB  ; test for rsvd. fields            [rae02]
002311  013440  010000  002506          NOP                     %JNZRO  U.CMDA  ;  if err jump to err rtn            [rae02]
002312  013440  000000  102303          NOP                     %JMP    U.IMEX  ; then exit successfully             [rae02]
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.ATTN (U.PROC ATTENTION PROCESS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE QUEUES A CONTROLLER INTERNAL 'ATTENTION' PACKET TO
                                ; A SDI CONTROL BLOCK IF THE 'DATT' BIT IS SET IN THE SDI CONTROL
                                ; BLOCK STATUS. IF THE OP CODE OF THE DEDICATED PACKET IS NONZERO THEN
                                ; THE ROUTINE EXITS, OTHERWISE IT QUEUES THE PACKET AND SETS THE OP CODE.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;       R7              = OP.ATT
                                ;       BAR             POINTS TO SDI.AT+P.OPCD IN SDI CTL BLK                          ;[E121]
                                ;       CCODE           ZERO IF DRIVE ONLINE                                            ;[E121]
                                ;
                                ; OUTPUTS:
                                ;       ATTENTION OR GET STATUS PACKET QUEUED TO SDI CONTROL BLOCK PACKET LIST
                                ;       R0,R1 ARE USED AS A TEMPORARY REGISTERS
                                ;
                                ;;-


                                ; ***   ENTRY POINT #1 - INITIAL ATTENTION PROCESSING ***                              ;[E121]
                                ;       BAR points to P.OPCD of the MSCP packet @ SDI.AT(UBAR)                          ;[E121]
                                                                                                                        ;[E121]
002313  033767  010016  052314  U.ATTN: MOV\T    #OP.GST,R7      %TNZRO          ; IF NOT ONLINE THEN USE GET STATUS OP CODE;[E121]
002314  013440  003007  010000          MOV     R7,BUF                          ; STORE ATTN/GET STAT OP CODE           ;[E121]
002315  033450  000005  000000          MOV     UBAR,R10                        ; MAKE A COPY OF UBAR                   ;[E121]
                                                                                                                        ;[E121]
                                ;       The following lines were removed by RL, 8/14/87, because they make no           ;[E121]
                                ;       sense whatsoever - R2 is not set to anything in particular!!                    ;[E121]
                                                                                                                        ;[E121]
                                ;       ADD     #P.OPCD,R2,BAR  %CALL   S.LDR2  ; [mjt06]R2 now has opcode of packet at top of queue;[E121
                                ;       XOR     #OP.ONL,R2\N                    ; [mjt06] is it an ONLINE?              ;[E121]
                                ;       XOR     #OP.GUS,R2\N    %JZRO   1$      ; [mjt06] if so, branch/else is it a GET UNIT STATUS?;[E12
                                ;       NOP                     %JZRO   1$      ; [mjt06] is so, branch to queue at top ;[E121]
                                ;                                                                                       ;[E121]
                                ;       ...     (rest of U.ATTN goes here)                                              ;[E121]
                                ;                                                                                       ;[E121]
                                ;1$:    ADD     #SDI.AT,R10     %JMP    U.LNKH  ; [mjt06]for ONLINE and GUS/go put the packet at the top o
                                                                                                                        ;[E121]
002316  030550  000046  000000          ADD     #SDI.AT,R10                     ; GO QUE PKT            [rae11]         ;[E121]
002317  010545  007022  125574          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LDR1 ; R1 => head of packet list             ;[E121]
002320  135541  000300  000000          BIC     #Q.STAT,R1                      ; Is packet list empty?                 ;[E121]
002321  113570  013200  142334          BIS\T   #PSTAT,R10\N,BUF %JZRO  20$     ; If so, put packet at head             ;[E121]
002322  010545  007000  135575          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR2                                         ;[E121]
002323  014542  000200  000000          BIT     #XCMP,R2                        ; If we are in the process of           ;[E121]
002324  013440  010000  102331          NOP                     %JZRO   10$     ; completing a command, and there       ;[E121]
002325  010545  007025  125575          ADD     #SDI.OM,UBAR\N,BAR %CALL S.LDR2 ; is an overlapped command as well,    ;[E121]
002326  013440  017001  025574          MOV     R1,BAR          %CNZRO  S.LDR1  ; then put the Attn command in back     ;[E121]
                                                                                ; of the overlapped command to prevent  ;[E121]
000072                          OVLPER  :=      72              ; a mismatch between the SDI.ST value   ;[E121]
                                                                ; (which probaby has BFRQ set) and      ;[E121]
```



```
                                                                                ; the opcode and args of the Attn packet;[E121]
002327  135541  000300  000000          BIC     #Q.STAT,R1                      ; Is there really an ovlp cmd there?    ;[E121]
002330  003760  010072  150027          MOV\T   #OVLPER,TSTCNT  %JZRO   ERRB1Z  ; No - Something is screwy...           ;[E121]
002331  013440  007001  125574  10$:    MOV     R1,BAR          %CALL   S.LDR1  ; Get link word of first/2d packet      ;[E121]
002332  113550  003300  010000          BIS     #<PSTAT+PACTV>,R10\N,BUF        ; STUFF THE CURRENT POINTER WHERE THE OLD POINTER WAS;[E12
002333  013440  007010  105620          MOV     R10,BAR         %JMP    S.STR1  ; SET THE ADDRESS OF FOR THE NEXT POINTER AND STUFF THE OL
                                                                                                                        ;[E121]
002334  013440  007010  000000  20$:    MOV     R10,BAR                         ; [mjt06] SET THE ADDRESS FOR THE NEXT POINTER;[E121]
002335  113740  003200  127777          MOV     #PSTAT,BUF      %RET            ; [mjt06] NO MORE PACKETS AFTER THIS ONE;[E121]

                                ; ***   ENTRY POINT #2 - ACTIVE PACKET PROCESSING ***

002336  133746  000100  135476  U.ATTA: MOV     #ERRIP,R6       %CALL   U.GMST  ; GET MUTEXED SDI STATUS IN R0
002337  014540  000200  010000          BIT     #XCMP,R0                        ; IF TRANSFER NOT COMPLETE
002340  035560  010040  152376          BIC\T   #SUSP,R0        %JZRO   U.CLSX  ;   THEN START PKT/FREE D.PROC/RETURN

                                ; ***   ENTRY POINT #3 - TERMINATION PROCESSING ***

002341  133544  000200  123711          BIS     #PLOCK,RLL      %CALL   U.ULKA  ; GO UNLINK PKT
002342  010545  007054  135630          ADD     #<SDI.AT+P.OPCD>,UBAR\N,BAR %CALL S.CLRB; ZAP OP CODE
002343  034446  000006  103645          CLR     R6              %JMP    U.CSTA  ; [U52EC1]GO CLEAR SDI STATUS/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.AVAL  (MSCP AVAILABLE COMMAND)
                                ;               (MSCP TOPOLOGY COMMAND)
                                ;       'SEQUENTIAL COMMANDS'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'AVAILABLE' AND 'TOPOLGY'
                                ; COMMAND.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET (1ST,2ND, AND 3RD ENTRY)
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (2ND AND 3RD ENTRY)
                                ;       R1              PACKET STATUS
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET QUEUED TO THE HOST.
                                ;
                                ;;-
020000                          M.AVAH  :=      MD.CSE                                  ; OP.AVL modifiers                      [rae02]
000043                          M.AVAL  :=      MD.EXC!MD.SPD!MD.ALL                    ; OP.AVL modifiers              [ch10][rae02]
                                U.AVAL:                                                                                 ;[rae02]
002344  003740  000043  000000          MOV     #M.AVAL,Q                               ; Set legal modifier bits       [rae02]
002345  103640  000040  102347          BIS     #M.AVAH,Q       %JMP    U.AVLO  ; Set modifier bits             [rae02]

000000                          M.DAP   :=      0                                       ;OP.DAP modifiers               [rae02]
002346  003740  000000  010000  U.DAP:  MOV     #M.DAP,Q                                ; Set modifier bits             [rae02]

                                U.AVLO:                                                                                 ;[rae02]
002347  114541  000100  010000          BIT     #PACTV,R1                               ; IF PACKET ACTIVE              [rae02]
002350  033442  010000  012354          MOV     R0,R2           %JNZRO  U.AVL1  ;   THEN GO PROCESS             [rae02]

                                ; ***   ENTRY POINT #1 - INITIAL PACKET PROCESSING ***

002351  013440  000000  122263          NOP                             %CALL   TSTMOD  ; test for reserved modifier bits       [rae02]
002352  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error         [rae02]
002353  013440  000000  103436          NOP                             %JMP    U.ONLA  ; do the rest at ONLINE         [rae02]

                                ; ***   ENTRY POINT #2 - ACTIVE PACKET PROCESSING ***

                                U.AVL1:                                                                                 ;[rae02]
002354  010545  007000  125574          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR1 ; R1=SDI STATUS
002355  014541  000200  000000          BIT     #XCMP,R1                                ; IF TRANSFER COMPLETE
002356  036542  010013  012362          XOR     #OP.DAP,R2      %JNZRO  U.AVLA  ;   THEN DONE
002357  114541  010020  012365          BIT     #DRAVL,R1       %JNZRO  U.AVLB  ; IF NOT ACCESS PATHS CMD THEN CONTINUE
002360  033740  010004  013641          MOV     #ST.AVL,R0      %JNZRO  U.DONE  ; IF AVAILABLE THEN DONE
002361  013440  000000  112373          NOP                             %JMP    U.AVLC  ;   ELSE CONTINUE

                                ; ***   DONE - CHECK FOR ACCESS PATHS COMMAND ***

002362  010545  017026  013640  U.AVLA: ADD     #SDI.ES,UBAR\N,BAR %JNZRO U.DONX; IF NOT ACCESS PATHS THEN DONE
002363  033702  000003  010000          MOV     (BUF),R2                                ; R2=EXTENDED STATUS
002364  033542  003004  103640          BIS     #FSEEK,R2,BUF   %JMP    U.DONX  ; SET FORCE SEEK FLAG/DONE
002365  010545  007012  135632  U.AVLB: ADD     #SDI.CH,UBAR\N,BAR %CALL INITM1 ; [V05] FFFF IN HI CYL ADDR
```



```
002366  010545  007044  135573          ADD     #SDI.S1,UBAR\N,BAR %CALL S.LDRO ; [mjt06] GET DRIVE RUN/STOP STATUS
002367                                  ASSUME DRV.RU,EQ,BIT00
002367  033760  040003  153641          MOV\T   #<ST.OFL>,R0    %JNLSB  U.DONE  ; [mjt06] IF SET/RUN/STOP IS OUT SO DRIVE IS OFFLINE AND DIS
002370  010550  007007  135573          ADD     #P.MOD,R10\N,BAR %CALL  S.LDRO  ; R0=MSCP MODIFIERS FIELD
002371                                  ASSUME  MD.SPD,EQ,BIT00                 ; MAKE SURE MD.SPD IS LSB
002371  114541  040020  002373          BIT     #DRAVL,R1       %JLSB   U.AVLC  ; IF DRIVE NOT ALREADY AVAILABLE
002372  033760  010000  043641          MOV\T   #ST.SUC,R0      %JNZRO  U.DONE  ; IF NO SPIN DOWN THEN DONE
002373  033746  000040  122610  U.AVLC: MOV     #SUSP,R6        %CALL   U.GUCP  ; GET UNIT CHAR PTR, SET R6 FOR U.CLRS
002374  033760  010003  153641          MOV\T   #ST.OFL,R0      %JZRO   U.DONE  ; IF NOT FOUND THEN DONE

                                ; ***   ROUTINE TO CLEAR SDI STATUS WITH BITS IN R6 ***

002375  013440  000000  125476  U.CLRS: NOP                             %CALL   U.GMST  ; GET MUTEXED SDI STATUS IN R0
002376  035140  003006  115474  U.CLSX: BIC     R6,R0,BUF       %JMP    UNLOCK  ; CLEAR APPROPRIATE BITS/FREE D.PROC/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.C2ED  (CHANGE TO END PACKET WITH DUPLICATE CHECK)
                                ;       U.C2EP  (CHANGE TO END PACKET)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE SETS THE MSCP 'END' PACKET CODE IN THE PACKET POINTED TO
                                ; BY 'SDI.PQ' IN THE SDI CONTROL BLOCK POINTED TO BY UBAR. IT ALSO SETS THE
                                ; CONDITION CODE IN R0 IN THE SAME PACKET. THE PACKET ADDRESS IS ENTERED IN
                                ; THE CONTROLLER TO HOST PACKET RING BUFFER.
                                ;
                                ; INPUTS:
                                ;       R0              CONDITION CODES FOR END PACKET
                                ;       R10             POINTER TO MSCP PACKET
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       CONVERTED MSCP PACKET
                                ;       UPDATED CONTROLLER TO HOST PACKET RING BUFFER
                                ;       R1 USED AS TEMPORARY REGISTER
                                ;       R2 USED AS TEMPORARY REGISTER
                                ;
                                ;;-

002377                                  ASSUME  ST.SUC,EQ,0                     ; MAKE SURE SUCCESS IS ZERO
002377  014540  000037  010000  U.C2ED: BIT     #ST.MSK,R0                      ; IF STATUS EQ 0 (SUCCESS)
002400  016540  010003  102413          XOR     #ST.OFL,R0\N    %JZRO   UC2EDA  ;   THEN CHK FOR DUP/ELSE IF EQ OFFLINE
002401  016540  010004  142402          XOR\F   #ST.AVL,R0\N    %TZRO           ;   THEN CHK FOR UNKNOWN/ELSE IF NOT AVAILABLE
002402  013440  010000  002420          NOP                     %JNZRO  U.C2EP  ;   THEN CONTINUE
002403  034442  000002  122477          CLR     R2              %CALL   U.CUNK  ; ZAP R2/CHECK FOR UNKNOWN
002404                                  ASSUME  SC.UNK,EQ,0                     ; MAKE SURE UNKNOWN IS ZERO
002404  106640  010024  102420          XOR     #<DRAVL!DRDUP>,Q %JZRO  U.C2EP  ; IF UNKNOWN THEN DONE
                        ;               BIS\T   #SC.IOP,R0      %JZRO   U.C2EP  ; [U52EC2]IF INOPERATIVE THEN SET SUBCODE/DONE
002405  010545  007044  125574          ADD     #SDI.S1,UBAR\N,BAR %CALL S.LDR1 ; R1=SDI STATUS
002406                                  ASSUME  DRV.RU,EQ,BIT00                 ; MAKE SURE DRV.RU IS LSB
002406  033562  040040  152407          BIS\T   #SC.NVL,R2      %TNLSB          ; IF RUN/STOP SWIT OUT THEN SET NO VOL MOUNTED FLG
002407  114541  000010  010000          BIT     #DRV.DD,R1                      ; IF DISABLED BY DIAGNOSTICS
002410  133562  010001  042411          BIS\T   #SC.DIS,R2      %TNZRO          ;   THEN DRIVE OFFLINE AND DISABLED
002411  003442  000002  010000          MOV     R2,Q                            ; IF R2 NEQ 0
002412  033660  010003  042413          BIS\T   #ST.OFL,Q,R0    %TNZRO          ;   THEN SET IN FLAGS CHANGE STATUS TO OFFLINE

                                ; ***   CHECK FOR DUPLICATE UNIT NUMBERS ***

002413  010545  007000  125574  UC2EDA: ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR1 ; R1=SDI STATUS
002414  114541  000004  010000          BIT     #DRDUP,R1                       ; IF DRIVE A DUPLICATE
002415  033560  010200  052416          BIS\T   #SC.DUP,R0      %TNZRO          ;   THEN OR IN DUPLICATE SUB-CODE
002416  013440  000000  102420          NOP                     %JMP    U.C2EP  ; CONTINUE
002417  033540  000003  123210  U.C2EZ: BIS     #ST.OFL,R0      %CALL   U.SBCN  ; R0=OFFLINE STATUS/ZAP BYTE COUNT      [rae03]
002420  010550  007007  135616  U.C2EP: ADD     #P.STS,R10\N,BAR %CALL  S.STRO  ; SET END PKT STATUS
002421  010550  007001  135573          ADD     #P.VCID,R10\N,BAR %CALL S.LDRO  ; R0=VIRTUAL CIRCUIT ID
002422  035540  000017  000000          BIC     #LONIB,R0                       ; ISOLATE CREDIT FIELD
002423  033540  003001  010000          OR      #1,R0,BUF                       ; CREDIT=1

                                        ;;CLEAR MAPPING FLAG
```



```
002424  114544  000100  010000          BIT     #DMODE,RLL                      ; DON'T STOMP DM STUFF
002425  013440  010000  002426          NOP                     %JNZRO  1$      ; SKIP IF DM
                                ;  no longer clear it here (as immediate mode commands might clear it for an inprogress ;[E122]
                                ;  command).  it is now cleared at U.IOPN:                                              ;[E122]
                                ;;;     ADD     #MAP.ST,UBAR\N,BAR %CALL S.CLRB ; CLEAR QDA STATUS                    ;[E122]
002426  010550  007006  135574  1$:     ADD     #P.OPCD,R10\N,BAR %CALL S.LDR1  ; R1=OP CODE TO CONVERT
002427  033541  003200  103340          OR      #OP.END,R1,BUF  %JMP    U.QSND  ; OR IN END CODE/LINK PKT TO CONTROLLER-HOST Q
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.CNVT (CONVERT LBN,RBN,DBN,XBN TO CYL,GROUP,TRACK,SECTOR)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL CONVERT A LBN,RBN,DBN, OR XBN TO CYLINDER,
                                ; GROUP, TRACK, AND SECTOR USING VARIABLES PASSED BY THE CALLER AND
                                ; STORING THE RESULTS IN A CALLER SPECIFIED AREA.
                                ;       NOTE THAT IT IS THE CALLER'S RESPONSIBILITY TO OR IN THE HIGH
                                ; ORDER START LBN,RBN,XBN, OR DBN.
                                ; THE FOLLOWING GENERAL EQUATIONS ARE USED:
                                ;       CYLINDER        : V1 + QUO(V2/(C*(S*V3)))
                                ;       GROUP           : QUO(REM(V2/(C*(S*V3)))/(S*V3))
                                ;       TRACK           : QUO(REM(REM(V2/(C*(S*V3)))/(S*V3))/V3)
                                ;       SECTOR(START)   : REM(REM(REM(V2/(C*(S*V3)))/(S*V3))/V3)+V4
                                ;       SECTOR(INDEX)   : REM((SECTOR(START)+(O*GROUP))/(T+R))
                                ; WHERE:
                                ;       C       : GROUPS PER CYLINDER
                                ;       O       : GROUP OFFSET
                                ;       R       : RBN'S PER TRACK
                                ;       S       : TRACKS PER GROUP
                                ;       T       : LBN'S PER TRACK
                                ;       V1      : VARIABLE #1 (32 bits)                                         ;[E121]
                                ;                 FOR LBN : The hi cylinder number
                                ;                 FOR RBN : The hi cylinder number
                                ;                 FOR XBN : The hi cylinder number + the LBN
                                ;                           Size in cylinders
                                ;                 FOR DBN : The hi cylinder number + the LBN
                                ;                           Size in cylinders + the XBN Space
                                ;                           in cylinders
                                ;       V2      : VARIABLE #2 (32 bits)                                         ;[E121]
                                ;                 FOR LBN : The LBN desired - the starting LBN
                                ;                 FOR RBN : The RBN desired - the starting RBN
                                ;                 FOR XBN : The XBN desired - the starting XBN
                                ;                 FOR DBN : The DBN desired - the starting DBN
                                ;       V3      : VARIABLE #3 (16 bits)                                         ;[E121]
                                ;                 FOR LBN : LBN's per track
                                ;                 FOR RBN : RBN's per track
                                ;                 FOR XBN : Sectors per track (LBN's + RBN's)
                                ;                 FOR DBN : Sectors per track (LBN's + RBN's)
                                ;       V4      : VARIABLE #4 (16 bits)                                         ;[E121]
                                ;                 FOR LBN : 0
                                ;                 FOR RBN : LBN's per track
                                ;                 FOR XBN : 0
                                ;                 FOR DBN : 0
                                ;
                                ; INPUTS:
                                ;       R3              POINTER TO UNIT CHARACTERISTICS
                                ;       UBAR            POINTER TO RESULT STORAGE AREA
                                ;       CNVTV1          CONVERSION VARIABLE #1 (V1)
                                ;       CNVTV2          CONVERSION VARIABLE #2 (V2)
                                ;       CNVTV3          CONVERSION VARIABLE #3 (V3)
                                ;       CNVTV4          CONVERSION VARIABLE #4 (V4)
                                ;
```



```
                                ; OUTPUTS:
                                ;       CYLINDER,GROUP,TRACK,SECTOR NUMBERS STORED IN THE AREA POINTED
                                ;       TO BY UBAR
                                ;       UBAR SET TO R3                                                          ;[E121]
                                ;       R10 RESTORED FROM SDI.PQ(UBAR)                                          ;[E121]
                                ;
                                ;;-

                                ; ***   CALCULATE CYLINDER NUMBER ***

002430  013740  007044  135572  U.CNVT: MOV     #CNVTV3,BAR     %CALL   S.LDQQ  ; Q=VARIABLE #3                        ;[E121]
002431  013740  007042  135574          MOV     #CNVTV2,BAR     %CALL   S.LDR1  ; R1=VARIABLE #2 LO                     ;[E121]
002432  013740  007043  135573          MOV     #CNVTV2+1,BAR   %CALL   S.LDR0  ; R0=VARIABLE #2 HI                     ;[E121]
002433  010543  007066  125600          ADD     #SDI.TG,R3\N,BAR %CALL  S.LDR6  ; R6 = TRACKS/GROUP                     ;[E121]
002434  034546  000377  125562          AND     #LOBYT,R6       %CALL   MULT    ; R7 = TRACKS/GROUP * V3                ;[E121]
002435  003447  000007  010000          MOV     R7,Q                            ; Q=R7                                  ;[E121]
002436  033450  000007  010000          MOV     R7,R10                          ; R10=R7                                ;[E121]
002437  010543  007065  125600          ADD     #SDI.GC,R3\N,BAR %CALL  S.LDR6  ; R6=GROUPS/CYLINDER                    ;[E121]
002440  034546  000377  125562          AND     #LOBYT,R6       %CALL   MULT    ; R7=TRKS/CYLINDER * V3                 ;[E121]
                                                                                ; (= SECTORS/CYLINDER)                  ;[E121]
                                                                                                                        ;[E121]
                                ; ***   NOTE - ONLY LO ORDER 16 BITS OF SECTORS/CYLINDER ARE USED ***            ;[E121]
                                                                                                                        ;[E121]
002441  033442  000007  135547          MOV     R7,R2           %CALL   DIVD    ; R1=QUO, R0=REM/RETURN                 ;[E121]
002442  013740  007040  125600          MOV     #CNVTV1,BAR     %CALL   S.LDR6  ; R6=VARIABLE #1 (LO)
002443  013740  007041  125601          MOV     #CNVTV1+1,BAR   %CALL   S.LDR7  ; R7=VARIABLE #1 (HI)
002444  030141  000006  000000          ADD     R6,R1                           ; ADD TO LO CYL #
002445  130467  020007  152446          INC\T   R7              %TCRY           ; IF CARRY THEN INCR R7
002446  033442  000010  132472          MOV     R10,R2          %CALL   U.ST1U  ; R2=S*V3/STORE R1/UPDATE UBAR
002447  120445  007005  135625          INC     UBAR\0,UBAR,BAR %CALL   S.STR7  ; STORE HI CYLINDER #

                                ; ***   CALCULATE GROUP NUMBER ***

002450  033441  000000  135546          MOV     R0,R1           %CALL   DIVDO   ; R1=REM/R1=QUO, R0=REM
002451  033446  000001  122472          MOV     R1,R6           %CALL   U.ST1U  ; SAVE GROUP IN R6

                                ; ***   CALCULATE TRACK AND START SECTOR ***

002452  013740  007044  125575          MOV     #CNVTV3,BAR     %CALL   S.LDR2  ; R2=VARIABLE #3
002453  111140  000002  010000          SUB     R2,R0\N                         ; OPTIMIZATION FOR 1 TRACK/GROUP -      ;[E121]
002454  034461  020001  052456          CLR\T   R1              %JNCRY  10$     ; SKIP DIVIDE IF QUO = 0                ;[E121]
002455  033441  000000  135546          MOV     R0,R1           %CALL   DIVDO   ; R1=REM(V2/(C*(S*V3))/S*V3)/R1=QUO, R0=REM;[E121]
002456  013740  007045  135601  10$:    MOV     #CNVTV4,BAR     %CALL   S.LDR7  ; R7=VARIABLE #4                        ;[E121]
002457  030140  000007  122472          ADD     R7,R0           %CALL   U.ST1U  ; ADD TO START SECTOR/STORE TRACK #     ;[E121]
002460  120445  007005  135616          INC     UBAR\0,UBAR,BAR %CALL   S.STR0  ; BUMP ADDR TO STORE START SECT         ;[E121]

                                ; ***   CALCULATE SECTORS FROM INDEX (R0 = START_SECTOR #,R6 = GROUP #) ***

002461  010543  007074  135601          ADD     #SDI.GO,R3\N,BAR %CALL  S.LDR7  ; R7<15:8> = GROUP OFFSET               ;[E121]
002462  001140  000006  132473          SUBC    R6,R0,Q         %CALL   U.CSPT  ; Q = PREDECR'D STARTING SECTOR FOR MULT;[E121]
                                                                                ; (LESS 1 TO LEAD THE SECTOR WE WANT)   ;[E121]
                                                                                ; R2 = SECTORS/TRACK (LBN+RBN)          ;[E121]
002463  031547  000377  000000  20$:    SUBC    #377,R7                         ; Q = START SECTOR + (GROUP OFFSET) *   ;[E121]
002464  000046  020006  102463          ADD     R6,Q            %JCRY   20$     ;     (GROUP NUMBER) BY REPEATED ADDS   ;[E121]
```

```
002465  101062  030002  042465          SUB\T   R2,Q            %JNMSB  .       ; REDUCE Q MODULO R2                          ;[E121]
002466  000042  000002  000000          ADD     R2,Q                            ; AND CORRECT FOR SUBTRACT OVERSHOOT         ;[E121]
002467  013440  007005  125613          MOV     UBAR,BAR        %CALL   S.STQQ  ; STORE SECTORS FROM INDEX                   ;[E121]
002470  033445  000003  010000          MOV     R3,UBAR                         ; SET UBAR TO INPUT VALUE OF R3              ;[E121]
002471  010545  007022  105602          ADD     #SDI.PQ,UBAR\N,BAR %JMP S.LD10  ; RESTORE R10/RETURN

                                ; ***   ROUTINE TO STORE R1 @UBAR AND INCREMENT UBAR ***

002472  120445  007005  105620  U.ST1U: INC     UBAR\0,UBAR,BAR %JMP    S.STR1  ; STORE LO CYLINDER #/RETURN

                                ; ***   ROUTINE TO CALCULATE SECTORS PER TRACK ***

002473                          U.CSPT: ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE LBNMSK IS LO BYTE
002473  010543  007074  135604          ADD     #SDI.12,R3\N,BAR %CALL  S.LLBO  ; R0=LBN'S PER TRACK
002474  010543  007067  125575          ADD     #SDI.RT,R3\N,BAR %CALL  S.LDR2  ; R2=RBN'S PER TRACK
002475  034542  000077  000000          AND     #RBNMSK,R2                      ; ISOLATE RBN'S
002476  030142  000000  127777          ADD     R0,R2           %RET            ; R2=LBN'S/TRK+RBN'S/TRK/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.CUNK  (CHECK FOR UNKNOWN UNIT)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL EXAMINE THE DRVOL,DRAVL AND DRDUP BITS OF SDI.ST
                                ; TO DETERMINE IF A DRIVE IS KNOWN BUT OFFLINE OR UNKNOWN.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       CONDITION CODE  =       ZERO    DRIVE UNKNOWN
                                ;       CONDITION CODE  =       NONZERO DRIVE OFFLINE BUT KNOWN
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-
002477  010545  007000  000000  U.CUNK: ADD     #SDI.ST,UBAR\N,BAR              ; BAR-PTR TO SDI STATUS
002500  103740  000034  010000          MOV     #<DRVOL+DRAVL+DRDUP>,Q          ; Q=DRIVE STATE BITS
002501  005600  000003  137777          BIC     (BUF),Q         %RET            ; IF DRIVE UNKNOWN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.CMDE  (MSCP COMMAND ERROR)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL TAKE THE MSCP PACKET OFFSET PASSED IN R0,
                                ; SHIFT IT LEFT BY ST.SHF AND OR IT TO THE ST.CMD STATUS CODE.
                                ;
                                ; INPUTS:
                                ;       R1      MSCP PACKET OFFSET
                                ;
                                ; OUTPUTS:
                                ;       R0      ST.CMD + R0*ST.SUB (SUB-CODE MULTIPLIER)
                                ;
                                ;;-

002502                                  ASSUME  ST.SHF,EQ,7                     ; MAKE SURE ST.SHF IS 7
002502  073541  000200  125504  U.CMDE: OR\L    #<ST.CMD*128.>,R1 %CALL S.SWB1  ; [US2EC3][EC0#2]POSITION PACKET OFFSET TO SUB-CODE FIELD
002503  010550  007010  125630          ADD     #P.BCNO,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields           ;[E122]
002504  010550  007011  135630          ADD     #P.BCN1,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields           ;[E122]
002505  033440  000001  113641          MOV     R1,R0           %JMP    U.DONE  ; MOVE TO R0/FINISH UP

                                ; ***   UNQUEUED PACKET COMMAND ERROR ROUTINE ***

                                ;********
                                ; Enter here (U.CMDA:)for a "protocol error" to clear the endcode       [rae1-C119]
                                ; to only OP.END protocol errors include:                               [rae1-C119]
                                ;       invalid message length                                         [rae1-C119]
                                ;       invalid MSCP version                                           [rae1-C119]
                                ;       invalid opcode                                                 [rae1-C119]
                                ;       invalid modifier                                               [rae1-C119]
                                ;       invalid Unit Flag                                              [rae1-C119]
                                ;       invalid Controller Flag                                        [rae1-C119]
                                ;       invalid reserved field                                         [rae1-C119]
                                ;********                                                              [rae1-C119]
                                U.CMDA:                                                               ;[rae1-C119]
002506  033740  000200  000000          MOV     #OP.END,R0                      ; Set opcode to OP.END  [rae1-C119]
002507  010550  007006  125616          ADD     #P.OPCD,R10\N,BAR %CALL S.STRO  ;                       [rae1-C119]
002510  013440  000000  112513          NOP                     %JMP  U.CMDC  ; and jump to set field in error          ;[E121]
                                ;********                                                              [rae1-C119]
                                ; Enter here (U.CMDB:) for invalid fields that are not "protocol errors [rae1-C119]
                                ; These include:                                                       [rae1-C119]
                                ;       bad buffer descriptors                                         [rae1-C119]
                                ;       illegal byte counts                                            [rae1-C119]
                                ;********                                                              [rae1-C119]
                                U.CMDB:                                                               ;[rae1-C119]
002511  010550  007010  125630          ADD     #P.BCNO,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields           ;[E121]
002512  010550  007011  135630          ADD     #P.BCN1,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields           ;[E121]
                                U.CMDC:                                                               ;[E121]
002513  073541  000200  125504          OR\L    #<ST.CMD*128.>,R1 %CALL S.SWB1  ; [US2EC3][EC0#2]POSITION PACKET OFFSET TO SUB-CODE FIELD
002514  033440  000001  102420          MOV     R1,R0           %JMP    U.C2EP  ; GO CONVERT TO END PACKET
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GCST (MSCP GET COMMAND STATUS COMMAND)
                                ;       'IMMEDIATE COMMAND'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'GET COMMAND STATUS' COMMAND
                                ; FOR THE CONTROLLER.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET QUEUED TO THE HOST CONTAINING THE
                                ; COMMAND STATUS.
                                ;       COMMAND STATUS  - 0 = COMMAND DONE OR NOT FOUND
                                ;                         COMPLEMENTED P.LBN0+P.LBN1 = COMMAND FOUND OR ACTIVE
                                ;
                                ;;-
000000                          M.GCST  := 0                                    ;no modifiers allowed for Get Cont   [rae02]
                                U.GCST:
002515  013440  000000  132254          NOP                     %CALL   MLN20   ; test that msg leng is large enough [rae02]
002516  034441  010001  002506          CLR     R1              %JNZR0  U.CMDA  ;   if not declare error             [rae02]
002517  003740  000000  132263          MOV     #M.GCST,0       %CALL   TSTMOD  ; test for reserved modifier bits      [rae02]
002520  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZR0  U.CMDA  ;   it set then error               [rae02]
002521  034447  000007  132536          CLR     R7              %CALL   U.GREF  ; GO MATCH REFERENCE NUMBER
002522  013440  010000  112533          NOP                     %JZR0   U.GCSB  ; IF NO MATCH THEN DONE
002523  010542  007017  122535          ADD     #S.LBNH,R2\N,BAR %CALL  U.GCSC  ; GO GET COMPLEMENTED LBN IN R7
002524  135547  000200  010000          BIC     #BIT15,R7                       ; [16K]PREVENT 2**32-1
002525  010550  007005  125573          ADD     #P.RSO3,R10\N,BAR %CALL S.LDRO  ; R0=COMPARE INDICATOR
002526  135547  010040  142531          BIC\F   #20000,R7       %JZR0   U.GCSA  ; [16K]IF EQ 0 THEN CONTINUE
002527  114540  000001  000000          BIT     #BIT08,R0                       ; IF ON RETRY CYCLE
002530  135567  010140  052531          BIC\T   #60000,R7       %TNZR0          ; [16K]THEN ZAP 1 MORE BIT
002531  010550  007013  135625  U.GCSA: ADD     #P.CMS1,R10\N,BAR %CALL S.STR7  ; STORE HI STATUS
002532  010542  007016  132535          ADD     #S.LBNL,R2\N,BAR %CALL  U.GCSC  ; R7=COMPLEMENTED LO LBN NUMBER
002533  010550  007012  125625  U.GCSB: ADD     #P.CMS0,R10\N,BAR %CALL S.STR7  ; SAVE DONENESS VALUE
002534  013440  000000  102303          NOP                     %JMP    U.IMEX  ; CHANGE TO END PKT/EXIT

002535  037707  000003  127777  U.GCSC: COM     (BUF),R7        %RET            ; COMPLEMENT BUFFER INTO R7
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GREF (GET COMMAND REFERENCE NUMBER MATCH)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL MATCH THE OUTSTANDING REFERENCE NUMBER FIELD
                                ; OF THE PACKET POINTED TO BY R10 TO COMMAND REFERENCE NUMBER OF AN
                                ; ACTIVE MSCP PACKET.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP ABORT OR GET COMMAND STATUS PACKET
                                ;
                                ; OUTPUTS:
                                ;       ZERO CONDITION CODE = FAILURE
                                ;       NONZERO CONDITION CODE = SUCCESS
                                ;       R2              POINTER TO PACKET OF MATCHED REFERENCE NUMBER
                                ;       R3              PACKET STATUS AND LINK WORD
                                ;
                                ;;-

002536  010550  007010  135573  U.GREF: ADD     #P.RFNO,R10\N,BAR %CALL S.LDRO  ; RO=OUTSTANDING REF # LO
002537  010550  007011  135574          ADD     #P.RFN1,R10\N,BAR %CALL S.LDR1  ; R1=OUTSTANDING REF # HI
                        ;       ASSUME  PKTEND&1,EQ,0                           ; [16K]MAKE SURE PKTEND IS EVEN
002540  033742  000352  010000          MOV     #PKTEND&LOBYT,R2                ; [16K]R2=END OF PACKET LIST ADDRESS
002541  133542  000002  010000          BIS     #PKTEND&HIBYT,R2                ; [16K]CONSTRUCT PKTEND IN R2
002542  131542  017032  177777  U.GRFA: SUB\F    #<MCP.LN+P>,R2,BAR %RZRO        ; IF EQ 0 THEN FAILURE RETURN/SUB PKT LENGTH FROM R2
002543  033703  000003  000000          MOV     (BUF),R3                        ; R3=STATUS AND LINK WORD
002544                                  ASSUME  PSTAT,EQ,BIT15                  ; MAKE SURE PSTAT IS MSB
002544  010542  037002  135610          ADD     #P.CRFO,R2\N,BAR %CMSB  S.XORO  ; CHECK LO REF #'S FOR EQUALITY
002545  010542  017003  012550          ADD     #P.CRF1,R2\N,BAR %JNZRO U.GRFB  ;   IF NOT THEN CONTINUE
002546  016501  000003  010000          XOR     (BUF),R1\N                      ; IF HI REF #'S EQ
002547  010542  017001  137777          ADD     #P.VCID,R2\N,BAR %RZRO          ; [16K] THEN RETURN
002550  016542  000342  102542  U.GRFB: XOR     #PKTBUF,R2\N    %JMP    U.GRFA  ; IF NOT LAST PKT
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GSDI (MATCH UNIT NUMBER TO SDI CONTROL BLOCK)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL MATCH A UNIT NUMBER GIVEN IN A MSCP PACKET
                                ; TO THE PROPER SDI CONTROL BLOCK. IF NO MATCH CAN BE FOUND IT WILL
                                ; RETURN AN ERROR.
                                ;       Beacuse duplicates can exist, first a search is done for an ONLINE drive        [rae03]
                                ;       with matching unit #.  If none then a search returns the first that             [rae03]
                                ;       matches, and if it is a duplicate the search is declared UNSUCCESSFUL           [rae03]
                                ;       and the SC.DUP is returned in RO.  Otherwise SUCCESS or UNSUCCESS is            [rae03]
                                ;       returned with RO=0.                                                             [rae03]
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET
                                ;
                                ; OUTPUTS:
                                ;       RO              = SC.DUP if an offline duplicate unit found                     [rae03]
                                ;                       = 0 otherwise                                                   [rae03]
                                ;       R6              SUBUNIT MASK IN BITS 15 - 12
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (IF SUCCESSFUL)
                                ;       CONDITION CODE NONZERO  - UNSUCCESSFUL RETURN (BAD UNIT NUMBER)
                                ;       CONDITION CODE ZERO     - SUCCESSFUL RETURN
                                ;
                                ;;-
                                U.GSDI:                                         ;                                       [rae03]
002551  010550  007004  135575          ADD     #P.UNIT,R10\N,BAR %CALL S.LDR2  ;get target unit #                      [rae03]
                                U.GSDN:                                         ;entry point with R2 = unit #           [rae03]
002552  013440  000000  125522          NOP                     %CALL S.SDI1    ;UBAR = ptr to 1st SDI entry            [rae03]
                                U.GSDA:                 ;search each SDI entry for ONL matching unit                    [rae03]
002553  013440  000000  132572          NOP                     %CALL U.GSDW    ;   IF current matches unit #           [rae03]
                                        ;Q=compliment of <DRAVL!DRVOL!DRDUP>                                            [rae03]
002554  106640  010030  102556          XOR     #<DRAVL!DRVOL>,Q  %JZRO U.GSDB  ;     and is ONL or ONL+DUP             [rae03]
002555  034460  010000  177777          CLR\T   RO                      %RZRO   ;     THEN return "SUCCESS"             [rae03]
                                U.GSDB:                                         ;     ELSE try next SDI entry           [rae03]
002556  030545  000120  135532          ADD     #SDIB.L,UBAR            %CALL S.FBC1    ;   increment SDI adrs          [rae03]
002557  036140  000005  010000          XOR     UBAR,RO                         ;   and test for end                    [rae03]
002560  013440  010000  002553          NOP                     %JNZRO U.GSDA ;if not loop                              [rae03]
                                        ;search each SDI entry for any matching unit                                    [rae03]
002561  013440  000000  125522          NOP                     %CALL S.SDI1    ;UBAR = ptr to 1st SDI entry            [rae03]
                                U.GSDC:                                         ;search each SDI entry                  [rae03]
002562  013440  000000  132572          NOP                     %CALL U.GSDW    ;   IF current matches unit #           [rae03]
                                        ;Q=compliment of <DRAVL!DRVOL!DRDUP>                                            [rae03]
002563  114640  010004  112566          BIT     #DRDUP,Q                %JZRO U.GSDD    ;     THEN  IF it is NOT DUP    [rae03]
002564  034460  010000  067777          CLR\T   RO                      %RNZRO          ;           THEN return "SUC"   [rae03]
002565  033740  000200  127777          MOV     #SC.DUP,RO              %RET            ;           ELSE return "FAIL"  [rae03]
                                U.GSDD:                                         ;    ELSE try next SDI                  [rae03]
002566  030545  000120  135532          ADD     #SDIB.L,UBAR            %CALL S.FBC1    ;   increment SDI adrs          [rae03]
002567  036140  000005  010000          XOR     UBAR,RO                         ;   and test for end                    [rae03]
                                                                                ;   IF not end THEN loop               [rae03]
002570  034440  010000  042562          CLR\F   RO                      %JNZRO U.GSDC ;             ELSE clear SC.DUP value [rae03]
002571  003740  000001  127777          MOV     #1,Q                    %RET            ;                 and return "FAIL" [rae03]
                                ;                                                                                       [rae03]
```

```
                                ; U.GSDW: is a local subroutine that tests if the current SDI block matches              [rae03]
                                ; the unit number in R2.                                                                 [rae03]
                                ;       Inputs:                                                                          [rae03]
                                ;               UBAR                    : current SDI block adrs                         [rae03]
                                ;               R2                      : desired unit #                                 [rae03]
                                ;       Outputs:                                                                         [rae03]
                                ;               Condition Code  ZRO     : No Match                                       [rae03]
                                ;                               NZRO    : Match                                          [rae03]
                                ;               Q                       : compliment of DRAVL,DRVOL,DRDUP bits           [rae03]
                                ;                                                                                        [rae03]
                                U.GSDW:                                         ;                                        [rae03]
002572  133746  000020  010000          MOV     #DRV.U1,R6                      ;get mask for 1st subunit                [rae03]
002573  010545  007043  010000          ADD     #SDI.UN,UBAR\N,BAR              ;get current SDI unit #                  [rae03]
002574  033700  000003  000000          MOV     (BUF),R0                        ;                                        [rae03]
002575  033443  000000  010000          MOV     R0,R3                           ;R3 = subunit flags                      [rae03]
002576  103740  000034  010000          MOV     #<DRAVL!DRVOL!DRDUP>,Q          ;set mask for drive status bits          [rae03]
002577  010545  007000  000000          ADD     #SDI.ST,UBAR\N,BAR              ;get drive status bits                   [rae03]
002600  005600  000003  010000          BIC     (BUF),Q                         ;Q=compliment of drive status bits       [rae03]
                                                                                ;if UNKNOWN then return "NO MATCH"       [rae03]
002601  135540  010374  142607          BIC\F   #DRV.UM,R0      %JZRO   U.GSDY  ;get only unit #                         [rae03]
                                U.GSDX:                                         ;                                        [rae03]
002602  014143  000006  010000          BIT     R6,R3                           ;if subuit doesn't exist                 [rae03]
002603  016140  010002  112607          XOR     R2,R0\N         %JZRO   U.GSDY  ;  then return "NO MATCH"                [rae03]
002604  033760  010001  177777          MOV\T   #1,R0           %RZRO           ;if unit #'s match return "MATCH"        [rae03]
002605  073446  000006  000000          ROT\L   R6,R6                           ;go to next subunit mask                 [rae03]
002606  130440  030000  002602          INC     R0              %JNMSB  U.GSDX  ;if not last subunit then loop           [rae03]
                                U.GSDY:                                         ;  else                                  [rae03]
002607  034440  000000  127777          CLR     R0              %RET            ;    return "NOMATCH"                    [rae03]
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GUCP (GET SUBUNIT CHARACTERISTICS POINTER)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL MATCH THE UNIT NUMBER IN THE MSCP PACKET
                                ; POINTED TO BY R10 TO A SUBUNIT BUFFER STATUS WORD'S UNIT NUMBER
                                ; AND WILL RETURN THE POINTER OF THE SUBUNIT BUFFER.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       R3              EQUALS UBAR
                                ;       Q               SDI.CW CONTENTS
                                ;       BAR             POINTING TO SDI.CW
                                ;       CONDITION CODE ZERO     - UNSUCCESSFUL RETURN (BAD UNIT NUMBER)
                                ;       CONDITION CODE NONZERO  - SUCCESSFUL RETURN
                                ;
                                ;;-

002610  010550  007004  125574  U.GUCP: ADD     #P.UNIT,R10\N,BAR %CALL S.LDR1  ; R1=UNIT #
002611  010545  007056  125572          ADD     #SDI.CW,UBAR\N,BAR %CALL S.LDQQ ;[V05] Q=SUBUNIT BUFFER STATUS (SDI.CW)
002612  114640  000374  010000          BIT     #DRV.UM,Q                       ; [U52EC2]IF NOT USED
002613  135640  010374  112616          BIC     #DRV.UM,Q,R0    %JZRO   UGUCPC  ; IF NOT IN USE THEN CONT/ELSE ISOLATE UNIT #
002614  036140  000001  000000          XOR     R1,R0                           ; IF EQ DESIRED UNIT
002615  033463  010005  167777          MOV\T   UBAR,R3         %RZRO           ;[V05] THEN RETURN (CCODE NEQ 0)
002616  034440  000000  127777  UGUCPC: CLR     R0              %RTN            ;[V05]
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GUST (MSCP GET UNIT STATUS COMMAND)
                                ;       'IMMEDIATE COMMAND'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'GET UNIT STATUS' COMMAND
                                ; FOR ANY UNIT KNOWN TO THE CONTROLLER.
                                ;
                                ; INPUTS:
                                ;       R1              MSCP PACKET QUEUE AND STATUS WORD
                                ;       R10             POINTER TO MSCP PACKET (1ST, 2ND, AND 3RD ENTRY)
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (2ND AND 3RD ENTRY)
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET QUEUED TO THE HOST CONTAINING THE
                                ; UNIT STATUS.
                                ;
                                ;;-

                                ; ***   ENTRY POINT #1 - COMPLETE PACKET PROCESSING ***

020000                          M.GUSH  := MD.CSE                               ;modifiers allowed for OP.GUS          [rae02]
000001                          M.GUSL  := MD.NXU                               ;modifiers allowed for OP.GUS          [rae02]
                                U.GUST:
002617  003740  000001  000000 U.GUS:  MOV     #M.GUSL,Q                       ; set legal modifier bits              [rae02]
002620  103640  000040  122263         BIS     #M.GUSH,Q       %CALL   TSTMOD  ; test for reserved modifier bits      [rae02]
002621  033761  010005  042506         MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error                 [rae02]
002622  010550  007007  135573         ADD     #P.MOD,R10\N,BAR %CALL  S.LDRO  ; RO=PKT MODIFIERS
002623                                 ASSUME  MD.NXU,EQ,BIT00                 ; MAKE SURE MD.NXU IS LSB
002623  010550  047004  102632         ADD     #P.UNIT,R10\N,BAR %JNLSB UGUSTC ;   THEN GO FIND SDI CNTRL BLK

                                ; ***   HANDLE NEXT UNIT MODIFIER ***

002624  033702  000003  135630         MOV     (BUF),R2        %CALL   S.CLRB  ; R2=UNIT START/ZAP UNIT # IN CASE NONE FOUND
002625  013440  000000  122552 UGUSTA: NOP                     %CALL   U.GSDN  ; GO MATCH THIS UNIT #
002626  114542  010374  102631         BIT     #DRV.UM,R2      %JZRO   UGUSTB  ; IF FOUND THEN EXIT
002627  130442  010002  102625         INC     R2              %JZRO   UGUSTA  ; IF UNIT NOT OVERFLOW 10 BITS  THEN LOOP
002630  013440  000000  112632         NOP                     %JMP    UGUSTC  ;   ELSE NEXT UNIT NOT FOUND
002631  010550  007004  125621 UGUSTB: ADD     #P.UNIT,R10\N,BAR %CALL S.STR2  ; RESET UNIT NUMBER

                                ; ***   FIND UNIT IN SDI CONTROL BLOCK AND ACKNOWLEDGE ATTENTION MESSAGE ***

002632  010550  007010  125630 UGUSTC: ADD     #P.BCNO,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields          ;[E122]
002633  010550  007011  135630         ADD     #P.BCN1,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields          ;[E122]
002634  010550  007022  135630         ADD     #P.RS16,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields          ;[E122]
002635  010550  007023  125630         ADD     #P.RS17,R10\N,BAR %CALL S.CLRB  ; clear the byte count fields          ;[E122]
002636  013440  000000  122551         NOP                     %CALL   U.GSDI  ; GO GET SDI PTR                       ;[E122]
002637  033550  010003  042420         BIS\T   #ST.OFL,RO      %JNZRO  U.C2EP  ; IF UNKNOWN THEN DONE                 [rae03]
002640  033746  000000  132610         MOV     #ST.SUC,R6      %CALL   U.GUCP  ; R6=SUCCESS STATUS
002641  033760  010003  142377         MOV\T   #ST.OFL,RO      %JZRO   U.C2ED  ; IF NOT FOUND THEN UNIT OFFLINE
002642  114640  000010  000000         BIT     #DRV.AV,Q                       ; IF DRIVE AVAILABLE
002643  033766  010004  042644         MOV\T   #ST.AVL,R6      %TNZRO          ;   THE R6=AVAILABLE STATUS
```



```
                                ; ***   FILL IN TRACK SIZE ***

002644  000550  000023  000000         ADD     #<P.TRCK-1>,R10,Q               ; Q=START OF STORAGE AREA-1
002645                                 ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE
002645  010545  007074  123366         ADD     #SDI.12,UBAR\N,BAR %CALL STBBLG ;[V05] GET AND SAVE LBN'S/TRACK

                                ; ***   FILL IN GROUP SIZE ***

002646                                 ASSUME  P.GRP,EQ,P.TRCK+1               ; MAKE SURE SEQUENTIAL
002646  010545  007066  123366         ADD     #SDI.TG,UBAR\N,BAR %CALL STBBLG ;[V05] GET AND SAVE TRACKS/GROUP

                                ; ***   FILL IN CYLINDER SIZE ***

002647                                 ASSUME  P.CYLS,EQ,P.GRP+1               ; MAKE SURE SEQUENTIAL
002647  010545  007065  123366         ADD     #SDI.GC,UBAR\N,BAR %CALL STBBLG ;[V05] GET AND SAVE GROUPS/CYLINDER

                                ; ***   FILL IN UNIT HARDWARE/SOFTWARE VERSION NUMBERS ***

002650                                 ASSUME  P.USVR,EQ,P.CYLS+1              ; MAKE SURE SEQUENTIAL
002650  010545  007036  135573         ADD     #SDI.RS,UBAR\N,BAR %CALL S.LDRO ;[V05] [ECO#2]GET AND SAVE UNIT H/W & S/W VERSION #S ;[E12
002651  135540  000200  133370         BIC     #FDIAG,RO               %CALL STROLG ; clear the FD bit supplied over the SDI          ;[E12
                                ; ***   FILL IN RCT SIZE ***

002652                                 ASSUME  P.RCTS,EQ,P.USVR+1              ; MAKE SURE SEQUENTIAL
002652  010545  007077  123365         ADD     #SDI.RV,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND SAVE RCT SIZE IN LBN'S

                                ; ***   FILL IN RBNS AND COPIES ***

002653  010545  007067  125573         ADD     #SDI.RT,UBAR\N,BAR %CALL S.LDRO ;[V05] RO=RBN'S/TRACK
002654  034540  000077  010000         AND     #RBNMSK,RO                      ; ISOLATE RBN'S/TRACK
002655  010545  007034  135574         ADD     #SDI.RC,UBAR\N,BAR %CALL S.LDR1 ; R1=RCT COPIES
002656  134541  000017  010000         AND     #LONIB*256.,R1                  ; ISOLATE RCT COPIES
002657                                 ASSUME  P.RBNS,EQ,P.RCTS+1              ; MAKE SURE SEQUENTIAL
002657  033140  000001  133370         OR      R1,RO           %CALL   STROLG  ; MERGE THE TWO AND STORE IN END PKT
002660  033440  000006  122377         MOV     R6,RO           %CALL   U.C2ED  ; RO=MSCP STATUS/GO CHANGE TO END PKT

                                ; ***   FILL IN UNIT, MEDIA TYPE IDENTIFIER'S ***

002661  013440  000000  113702         NOP                     %JMP    U.SUNI  ; SET IN UNIT,MEDIA TYPE ID'S/RETURN
                                       .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.IOPR (MSCP I/O PROCESS, READ,WRITE,COMPARE,ACCESS,ERASE)
                                ;       'NON-SEQUENTIAL COMMANDS'
                                ;       U.IOPO (the same but tests reserved fields first)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE FOLLOWING FUNCTIONS ON THE FIRST
                                ; ENTRY:
                                ;       1. MATCH UNIT NUMBER TO SDI CONTROL BLOCK (ERROR IF NO MATCH)
                                ;       2. QUE PACKET ON SDI CONTROL BLOCK'S PACKET LIST
                                ;       3. CALCULATE CYLINDER, GROUP, TRACK, SECTOR(START), SECTOR(INDEX)
                                ; ON THE SECOND ENTRY THE FOLLOWING FUNCTIONS ARE PERFORMED:
                                ;       1. PRESET THE NUMBER OF BYTES REMAINING FOR THIS TRANSFER
                                ;       2. SET THE NUMBER OF SECTORS FOR THE HEADER COMPARE SEARCH LIMIT
                                ;       3. PRESET THE CURRENT HOST MEMORY ADDRESS
                                ;       4. CHECKS THE NEW CYLINDER NUMBER AGAINST THE CURRENT CYLINDER
                                ;          NUMBER AND DOES NOT INITIATE A SEEK IF THEY ARE EQUAL
                                ;       5. CHECKS THE NEW GROUP NUMBER AGAINST THE CURRENT GROUP NUMBER
                                ;          AND DOES NOT ISSUE A SELECT GROUP COMMAND IF THEY ARE EQUAL
                                ;
                                ; INPUTS:
                                ;       R0              MSCP OP CODE
                                ;       R1              PACKET QUEUE AND STATUS WORD
                                ;       R10             MSCP PACKET POINTER (1ST, 2ND, AND 3RD ENTRY)
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (2ND AND 3RD ENTRY)
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET IF ERROR
                                ;       CALCULATED PARAMETERS AS DESCRIBED ABOVE
                                ;       UPDATED SDI STATUS TO 'INIT SEEK' OR 'BUFFER REQUEST' AS DETERMINED
                                ;       BY THE CYLINDER MATCH OUTCOME
                                ;
                                ;;-
127400                          M.CMPH  :=  MD.EXP!MD.CSE!MD.SCH!MD.SCL!MD.SEC!MD.SER   ;OP.CMP modifiers[rae02]
000200                          M.CMPL  :=  MD.SSH                                      ;OP.CMP modifiers[rae02]
002662  003740  000200  000000  U.CMPR: MOV     #M.CMPL,Q                       ; set legal modifier bits          [rae02]
002663  103640  000257  102705          BIS     #M.CMPH,Q       %JMP    U.IOPR  ; set legal P.MOD bits             [rae02]

167400                          M.RDH   :=  MD.EXP!MD.CMP!MD.CSE!MD.SCH!MD.SCL!MD.SEC!MD.SER        ;OP.RD modifiers [rae02]
000200                          M.RDL   :=  MD.SSH                                                  ;OP.RD modifiers [rae02]
002664  003740  000200  000000  U.RD:   MOV     #M.RDL,Q                        ; set legal modifier bits          [rae02]
002665  103640  000357  112705          BIS     #M.RDH,Q        %JMP    U.IOPR  ; set legal P.MOD bits             [rae02]

171400                          M.WRH   :=  MD.EXP!MD.CMP!MD.CSE!MD.ERR!MD.SEC!MD.SER               ;OP.WR modifiers [rae02]
000360                          M.WRL   :=  MD.SSH!MD.WBN!MD.WBV!MD.SEQ                             ;OP.WR modifiers [rae02]
002666  003740  000360  010000  U.WR:   MOV     #M.WRL,Q                        ; set legal modifier bits          [rae02]
002667  103640  000363  102705          BIS     #M.WRH,Q        %JMP    U.IOPR  ; set legal P.MOD bits             [rae02]

                                ; ***  ACC, ERS, RPL must verify reserved fields, hence call U.IOPO                [rae02]

127400                          M.ACCH  :=  MD.EXP!MD.CSE!MD.SCH!MD.SCL!MD.SEC!MD.SER               ;OP.ACC modifiers[rae02]
000200                          M.ACCL  :=  MD.SSH                                                  ;OP.ACC modifiers[rae02]
002670  003740  000200  000000  U.ACC:  MOV     #M.ACCL,Q                       ; set legal modifier bits          [rae02]
```



```
002671  103640  000257  102676          BIS     #M.ACCH,Q       %JMP    U.IOPO  ; set legal P.MOD bits             [rae02]

130400                          M.ERSH  :=  MD.EXP!MD.CSE!MD.ERR!MD.SER                             ;OP.ERS modifiers[rae02]
000360                          M.ERSL  :=  MD.SSH!MD.WBN!MD.WBV!MD.SEQ                             ;OP.ERS modifiers[rae02]
002672  003740  000360  010000  U.ERS:  MOV     #M.ERSL,Q                       ; set legal modifier bits          [rae02]
002673  103640  000261  102676          BIS     #M.ERSH,Q       %JMP    U.IOPO  ; set legal P.MOD bits             [rae02]

120000                          M.RPLH  :=  MD.EXP!MD.CSE                                           ;OP.RPL modifiers[rae02]
000001                          M.RPLL  :=  MD.PRI                                                  ;OP.RPL modifiers[rae02]
002674  003740  000001  000000  U.RPL:  MOV     #M.RPLL,Q                       ; set legal modifier bits          [rae02]
002675  103640  000240  010000          BIS     #M.RPLH,Q                       ; set legal P.MOD bits             [rae02]

                                U.IOPO:
002676  114541  000100  010000          BIT     #PACTV,R1                       ; IF PACKET ACTIVE                 [rae02]
002677  013460  010000  042770          NOP\T                   %JNZRO  U.IOPC  ;   THEN GO TO ENTRY POINT #2     [rae02]
002700  013440  000000  122263          NOP                     %CALL   TSTMOD  ; Test modifier bits               [rae02]
002701  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error            [rae02]
002702  013440  000000  122247          NOP                     %CALL   U.RSVB  ; Test reserved fields             [rae02]
002703  013440  010000  002506          NOP                     %JNZRO  U.CMDA  ;   if error goto error rtn       [rae02]
002704  013440  000000  112711          NOP                     %JMP    U.IOP1  ; join common code                 [rae02]
                                U.IOPR:
002705  114541  000100  010000          BIT     #PACTV,R1                       ; IF PACKET ACTIVE
002706  013460  010000  042770          NOP\T                   %JNZRO  U.IOPC  ;   THEN GO TO ENTRY POINT #2

                                ; ***   ENTRY POINT #1 - INITIAL PACKET PROCESSING ***

002707  013440  000000  122263          NOP                     %CALL   TSTMOD  ; Test modifier bits               [rae02]
002710  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error            [rae02]
                                U.IOP1:                                         ;                                  [rae02]
002711  013440  000000  122255          NOP                     %CALL   MLN40   ; test that msg leng is large enough [rae02]
002712  034441  010001  002506          CLR     R1              %JNZRO  U.CMDA  ;   if not declare error         [rae02]
002713  034441  000001  122551          CLR     R1              %CALL   U.GSDI  ;   ELSE 1ST ENTRY/GO FIND SDI CONTROL BLK
002714  034462  010002  052417          CLR\T   R2              %JNZRO  U.C2EZ  ; IF OFFLINE THEN DONE
002715  034442  000002  122610          CLR     R2              %CALL   U.GUCP  ; GET THIS UNIT CHAR PTR IN R3
002716  034461  010001  173210          CLR\T   R1              %CZRO   U.SBCN  ; IF OFFLINE ZAP BYTE COUNT
002717  033760  010003  142377          MOV\T   #ST.OFL,R0      %JZRO   U.C2ED  ; IF OFFLINE THEN FIND OUT WHY
002720                                  ASSUME  P.LBN1,EQ,P.LBNO+1              ; MAKE SURE SEQUENTIAL
002720  000550  007021  133211          ADD     #P.LBN1,R10,Q,BAR %CALL LDR2R1  ; R1/R2=LO/HI MSCP LBN
002721  114542  000360  000000          BIT     #HDCOD,R2                       ; IF LBN OVERFLOW
002722  033761  010016  052511          MOV\T   #<P.LBNO-P>,R1  %JNZRO U.CMDB   ;    report error       [rae1-C119]
                                10$:
                                ; ***   CHECK IF START LBN GR/EQ 1ST RCT LBN ***

002723  133740  000002  133215          MOV     #SECSZ*2,R0     %CALL   U.CRCT  ; [16K]IF LBN GR/EQ 1ST RCT LBN
002724  031440  020000  102756          DEC     R0              %JCRY   U.IOPB  ; [16K]  THEN OK/CONTINUE

                                ; ***   CHECK BYTE COUNT FOR GREATER THAN 22-BITS ***

002725  010550  007013  135576          ADD     #P.BUF1,R10\N,BAR %CALL S.LDR3  ; GET MAPPING INDICATOR WORD
002726  013440  030000  012732          NOP                     %JNMSB  U.IOPS  ; IF MAPPING NOT ACTIVE THEN SKIP THIS
002727  010550  007011  135572          ADD     #P.BCN1,R10\N,BAR %CALL S.LDQQ  ; CHECK BYTE COUNT SIZE
002730  114640  000300  010000          AND     #BANOT,Q\N                      ; [BDA] CHECK HIGH BITS
002731  033761  010006  042511          MOV\T   #<P.BCNO-P>,R1  %JNZRO  U.CMDB  ; ANY BIT ABOVE BIT21 THEN ERROR [rae1-C119]
                                U.IOPS:                                         ;
```

```
                                ; ***   CONVERT BYTE COUNT TO BLOCK COUNT ***

002732  010550  007010  125572            ADD     #P.BCNO,R10\N,BAR %CALL S.LDQQ  ; [16K]Q=LO BYTE COUNT
002733  010550  007011  125601            ADD     #P.BCN1,R10\N,BAR %CALL S.LDR7  ; [16K]R7=HI BYTE COUNT
002734  013047  000007  010000            OR      Q,R7\N                          ; [16K]IF BYTE CONT EQ 0
002735  000040  010000  102756            ADD     R0,Q            %JZRO   U.IOPB  ; [16K]  THEN SKIP CHK/ELSE ADD 777 TO LO BYTE CNT
002736  130467  020007  152737            INC\T   R7              %TCRY           ; [16K]IF CARRY THEN INCR R7
002737  033740  060011  012744            MOV     #9.,R0          %JNUPF  U.IOPA  ; [16K]IF NOT REPLACE THEN R0=LOOP COUNT;[E121]
002740  010545  007067  125573            ADD     #SDI.RT,UBAR\N,BAR %CALL S.LDR0 ; Replace - get RBN's/track                     ;[E121]
002741  034540  000077  010000            AND     #RBNMSK,R0                                                                      ;[E121]
002742  033741  010006  112506            MOV     #<P.RBNO-P>,R1  %JZRO   U.CMDA  ; Illegal if no RBN's! (SSD)                   ;[E122]
002743  013440  000000  112756            NOP                     %JMP    U.IOPB  ; Otherwise skip block extent test              ;[E121]
002744  043447  010007  152747  U.IOPA:   ROT\R0F R7               %JZRO   U1OPA1  ; [16K]DOUBLE RIGHT SHIFT R7,Q
002745  103660  040200  042746            BIS\T   #BIT15,Q        %TLSB           ; [16K]IF LSB OF R7 THEN SET MSB OF Q
002746  031440  000000  112744            DEC     R0              %JMP    U.IOPA  ; [16K]DECR LOOP COUNT/LOOP

                                ; ***   ADD BLOCK COUNT IN Q AND R7 TO R1 AND R2 (START LBN) ***

002747  034547  000177  010000  U1OPA1:   AND     #177,R7                         ; [16K]ISOLATE LIVE BITS
002750  001240  000000  000000            DEC     Q                               ; [16K]DECR Q
002751  031467  020007  042752            DEC\T   R7              %TNCRY          ; [16K]AND R7 IF BORROW
002752  030041  000001  000000            ADD     Q,R1                            ; [16K]ADD LO BLK CNT TO LO LBN
002753  130462  020002  152754            INC\T   R2              %TCRY           ; [16K]IF CARRY THEN INCR HI LBN
002754  030142  000007  133215            ADD     R7,R2           %CALL   U.CRCT  ; [16K]ADD HI BLK CNT TO HI LBN
002755  033761  020006  152511            MOV\T   #<P.BCNO-P>,R1  %JCRY   U.CMDB  ; [16K]IF GR/EQ RCT THEN ERROR [rae1-C119]
002756  010545  007022  133222  U.IOPB:   ADD     #SDI.PQ,UBAR\N,BAR %CALL U.LNKP ; [16K]GO LINK PKT TO LIST

                                ; ***   COMPARE MODIFIER OR UNIT FLAG RE-ENTRY POINT (CALLED FROM U.IOPJ) ***

002757                          U.IOCM:   ASSUME  P.LBN1,EQ,P.LBNO+1              ; MAKE SURE SEQUENTIAL
002757  000550  007021  133211            ADD     #P.LBN1,R10,Q,BAR %CALL LDR2R1  ; R1/R2=LO/HI MSCP LBN
002760                                    ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE
002760  010545  007074  135604            ADD     #SDI.12,UBAR\N,BAR %CALL S.LLBO ;[V05] GET LBN'S PER TRACK IN R0<7:0>

                                ;       REVECTOR CODE ENTERS HERE TO CONVERT RCT BLOCK NUMBER
                                ;       LBN/RBN IN (R2,R1),LBN/RBN PER TRACK IN R0, //R3-->SUBUNIT CHAR, R10-->MSCP PKT

002761  010545  007064  125601  U.IOPX:   ADD     #SDI.H2,UBAR\N,BAR %CALL S.LDR7 ;[V05] R7 \<15:12> = HI CYL NUMBER
002762                                    ASSUME  <CNVTV1&17>,LE,<17-<CNVTV4-CNVTV1>>; USE AUTO-INC TO PAINT CNVTVX
002762  013740  007040  125630            MOV     #CNVTV1,BAR     %CALL   S.CLRB  ; LOW START CYL NUMBER = 0
002763  134547  003360  125620            AND     #CYLSTR,R7,BUF  %CALL   S.STR1  ; MASK & STORE HI START CYL, LOW BLOCK #

                                ;       ONLINE CODE ENTERS HERE TO CALCULATE FIRST XBN CYLINDER ***
                                ;       HI XBN IN R2, R10->MSCP PACKET

002764  013440  003002  125616  U.IOPY:   MOV     R2,BUF          %CALL   S.STRO  ; STORE HI BLOCK #, BLOCKS/TRACK
002765  030550  000024  135630            ADD     #S.CYLL,R10     %CALL   S.CLRB  ; SECTOR OFFSET = 0, POINT R10 AT RESULT BUF
002766  033443  000005  010000            MOV     UBAR,R3                                                                         ;[E121]
002767                                    ASSUME  S.CYLH,EQ,S.CYLL+1                                                              ;[E121]
002767                                    ASSUME  S.GRUP,EQ,S.CYLH+1                                                              ;[E121]
002767                                    ASSUME  S.TRAK,EQ,S.GRUP+1                                                              ;[E121]
002767                                    ASSUME  S.SECS,EQ,S.TRAK+1                                                              ;[E121]
002767  033445  000010  112430            MOV     R10,UBAR        %JMP    U.CNVT  ; UBAR=RESPONSE AREA/GO CNVT ?BN/RETURN
```



```
                                                                ; **NOTE** IF RBN, SECTOR POS. WILL BE WRONG

                                ; ***   ENTRY POINT #2 - ACTIVE PACKET PROCESSING ***
                                ; ***   @RUPF IF NORMAL I/O, @SUPF IF REPLACE COMMAND ***
                                ; ***   NOTE - R0 MUST HAVE OP CODE !!! ***

002770  033447  000000  122610  U.IOPC:   MOV     R0,R7           %CALL   U.GUCP  ; [16K]R7=OPCODE
002771  033760  010003  153022            MOV\T   #ST.OFL,R0      %JZRO   UIOPER  ; IF OFFLINE THEN EXIT                          ;[E121]
002772  010545  007047  125630            ADD     #SDI.2T,UBAR\N,BAR %CALL S.CLRB ;[V05] [ECO#2]CLR TEMP #2                       ;[E121]
002773  134642  000010  000000            AND     #DRV.AV,Q,R2                    ; IF SUBUNIT AVAILABLE                          ;[E121]
002774  033760  010004  053022            MOV\T   #ST.AVL,R0      %JNZRO  UIOPER  ;   THEN DONE                                   ;[E121]
002775  010545  007000  135573            ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR0 ; R0=SDI CNTRL BLK STATUS
002776  014540  000200  010000            BIT     #XCMP,R0                        ; IF TRANSFER COMPLETE
002777  133741  010002  013110            MOV     #SECSZ*2,R1     %JNZRO  U.IOPG  ; R1=WORD COUNT/R2=0 (IN CASE REPLACE)
003000                                    ASSUME  P.BCN1,EQ,P.BCNO+1              ; MAKE SURE SEQUENTIAL
003000  000570  067011  063211            ADD\T   #P.BCN1,R10,Q,BAR %CNUPF LDR2R1 ; IF NOT REPLACE THEN R1/R2=BYTE COUNT LO/HI
003001  033760  040111  053022            MOV\T   #<ST.HST+SC.ODB>,R0 %JLSB UIOPER; IF ODD THEN ODD BYTE COUNT BUFFER ERROR;[E121]
003002                                    ASSUME  SDI.XH,EQ,SDI.XL+1              ; MAKE SURE SEQUENTIAL
003002  000545  007015  123213            ADD     #SDI.XL,UBAR,Q,BAR %CALL STR1R2 ; SAVE LO/HI TRANSFER COUNT

                                ; ***   SET CURRENT HOST MEMORY ADDRESS ***

003003                                    ASSUME  <OP.RPL!OP.ACC!OP.ERS>&20,NE,0  ; [16K]MAKE SURE CHECK WILL WORK
003003  014547  000020  000000            BIT     #20,R7                          ; [16K]IF ACC,ERS,OR RPL
003004  013440  010000  013011            NOP                     %JNZRO  U.IOPW  ; [16K]  THEN SKIP BUFFER DESCRIPTOR
003005                                    ASSUME  P.BUFH,EQ,P.BUFL+1              ; MAKE SURE SEQUENTIAL
003005  000550  007013  123211            ADD     #P.BUFH,R10,Q,BAR %CALL LDR2R1  ; R1/R2=HOST ADDR LO/HI
003006  033760  040051  053022            MOV\T   #<ST.HST+SC.ODT>,R0 %JLSB UIOPER ; IF ODD THEN ODD TRANSFER ADDR BUFFER ERROR;[E121]
003007  135542  000300  000000            BIC     #BANOT,R2                       ;[B0A] MAKE SURE RESERVED IS ZERO
003010                                    ASSUME  S.ADRH,EQ,S.ADRL+1              ; MAKE SURE SEQUENTIAL
003010  000550  007022  123213            ADD     #S.ADRL,R10,Q,BAR %CALL STR1R2  ; SAVE LO/HI HOST ADDR

                                ; ***   SET CURRENT HEADER ADDRESS AND CHECK CYLINDER AGAINST MAX ***

003011  010550  007024  125601  U.IOPW:   ADD     #S.CYLL,R10\N,BAR %CALL S.LDR7  ; [16K]R7=LO CALCULATED CYL #
003012  010550  007025  125606            ADD     #S.CYLH,R10\N,BAR %CALL S.LLB6  ; R6=HI CALCULATED CYL #
003013  010545  007063  135575            ADD     #SDI.H1,UBAR\N,BAR %CALL S.LDR2 ;[V05] R2=LO CYLS IN HOST AREA
003014  010545  007064  125604            ADD     #SDI.H2,UBAR\N,BAR %CALL S.LLBO ;[V05] R0=HI CYLS IN HOST AREA
003015  132140  000006  010000            RSUB    R6,R0                           ; SUBTRACT HI CYL #'S
003016  112162  010007  143017            CMP\T   R7,R2           %TZRO           ; IF EQ 0 THEN COMPARE LO CYL #'S
003017  033741  020016  043032            MOV\F   #<P.LBNO-P>,R1  %JNCRY  U.IOPP  ; [U52EC1]IF NO CRY THEN OK & CONTINUE
003020  010545  007025  125573            ADD     #SDI.0M,UBAR\N,BAR %CALL S.LDR0 ; [U52EC1]  ELSE CHK FOR OVERLAPPED CMD
003021  013440  010000  102502            NOP                     %JZRO   U.CMDE  ; and report error                              [C118]
003022  010545  007025  135600  UIOPER:   ADD     #SDI.0M,UBAR\N,BAR %CALL S.LDR6 ; [U52EC1]R6=OVERLAPPED COMMAND                  ;[E121]
003023  013740  013000  153027            MOV\F   #0,BUF          %JZRO   10$     ; [U52EC1]IF NO THEN CONTINUE                    ;[E121]
003024  010545  007015  135630            ADD     #SDI.XL,UBAR\N,BAR %CALL S.CLRB ; [V05] CLEAR XL FOR OVERLAPPED COMMANDS
003025  010545  007016  135630            ADD     #SDI.XH,UBAR\N,BAR %CALL S.CLRB ; [V05] CLEAR XH FOR SAME
003026  033450  000006  103230            MOV     R6,R10          %JMP    U.LNKH  ; [U52EC1]  ELSE RESTORE OVERLAPPED CMD TO TOP OF QUE
003027  010545  007003  135616  10$:      ADD     #SDI.SW,UBAR\N,BAR %CALL S.STRO ; SAVE ERROR CODE                               ;[E121]
003030  034441  060001  143176            CLR\F   R1              %JUPF   U.IOPM  ; IF REPLACE THEN CLEANUP/EXIT
003031  034442  000002  103172            CLR     R2              %JMP    U.IOSA  ;   ELSE ZAP BYTE COUNT/EXIT
003032  010545  007013  125573  U.IOPP:   ADD     #SDI.GP,UBAR\N,BAR %CALL S.LDR0 ; R0=CURRENT GROUP NUMBER
003033  010545  007005  135616            ADD     #SDI.UG,UBAR\N,BAR %CALL S.STRO ; SAVE FOR U.PROC GROUP COMPARE
```

```
                                ; ***   ONLINE COMMAND ENTERS HERE TO FINISH I/O PREPARATION ***
                                ; ***   CYLINDER IN (R7,R6), R10 -> PKT, UBAR -> SDI BLK, //R3 -> SUBUNIT BLK ***
                                ; ***   RO HAS LBN/XBN ADJUSTMENT NIBBLE IN BITS <11:8> OF RO ***
                                ; ***   SAVE STARTING LBN LO AND HI IN S.LBNL AND S.LBNH WHICH OVERLAY S.CYLX ***

003034                          U.IOPZ: ASSUME  P.LBN1,EQ,P.LBNO+1              ; MAKE SURE SEQUENTIAL
003034  000550  007021  133211          ADD     #P.LBN1,R10,Q,BAR %CALL LDR2R1  ; R1/R2=LO/HI STARTING LBN
003035                                  ASSUME  LBNSTR*16.,EQ,XBNSTR            ; MAKE SURE MASKS ARE OK
                                ;       AND     #LBNSTR,RO                      ; [U52EC2] ISOLATE HI STARTING LBN
                                ;       ADD     RO,R2                           ; [U52EC2] ADD INTO HI LBN
003035                                  ASSUME  S.LBNH,EQ,S.LBNL+1              ; MAKE SURE SEQUENTIAL
003035  000550  007016  133213          ADD     #S.LBNL,R10,Q,BAR %CALL STR1R2  ; SAVE LO/HI LBN'S

                                ; ***   SET UP SDI OP CODE FROM MSCP OP CODE ***

003036  103740  000027  000000          MOV     #READCD,Q                       ; ASSUME OPERATION IS A READ
003037  010545  007022  134123          ADD     #SDI.PQ,UBAR\N,BAR %CALL CKWRER ; IF OP IS NOT WRITE, ERASE, OR REPLACE
003040  103740  010245  153044          MOV\F   #WRITCD,Q       %JZRO   10$     ;   THEN CONTINUE/ELSE CHANGE OP CODE   ;[E121]

                                ; ***   WRITE, ERASE, OR REPLACE COMMAND - CHECK FOR WRITE PROTECT ***

003041  010545  007060  135573          ADD     #SDI.UF,UBAR\N,BAR %CALL S.LDRO ; RO=UNIT FLAGS
003042                                  ASSUME  UF.WPH,EQ,SC.WPH                ; MAKE SURE BITS ARE THE SAME
003042                                  ASSUME  UF.WPS,EQ,SC.WPS                ; MAKE SURE BITS ARE THE SAME
003042  134540  000080  000000          AND     #<UF.WPH!UF.WPS>,RO             ; ISOLATE WRITE PROTECT FLAGS
003043  033540  010006  013022          OR      #ST.WPR,RO      %JNZRO  UIOPER  ; IF WRITE PROTECTED THEN OR IN BITS/DONE;[E121]

                                ; ***   CALCULATE SEARCH LIMIT FOR D.PROC ***

003044  013440  000000  132473  10$:    NOP                     %CALL   U.CSPT  ; [U52EC2]GO CALC SECTORS/TRK IN R2     ;[E121]
003045  010545  007023  135621          ADD     #SDI.RO,UBAR\N,BAR %CALL S.STR2 ; [U52EC2]AND IN ROTATIONAL OPTIMIZATION COUNT
003046  030142  000002  010000          ADD     R2,R2                           ; [U52EC2]SEARCH LIMIT IS NOMINALLY 2 REVS
003047                                  ASSUME  WRITCD&BIT15,NE,0               ; [U52EC2]
003047                                  ASSUME  READCD&BIT15,EQ,0               ; [U52EC2]
003047  013240  000000  000000          TST     Q                               ; [U52EC2]EXCEPT FOR READS
003050  130162  030000  043051          ADDC\T  RO,R2            %TNMSB         ; [U52EC2]WHICH GET AN EXTRA REV
003051  010545  007001  135621          ADD     #SDI.SL,UBAR\N,BAR %CALL S.STR2 ; SAVE IN SDI CNTRL BLK

                                ; ***   CHECK NEW CYLINDER AGAINST CURRENT CYLINDER ***

003052  010545  007064  125575          ADD     #SDI.H2,UBAR\N,BAR %CALL S.LDR2 ;[V05] R2=HI CYL #
003053  134542  000377  010000          AND     #HIBYT,R2                       ; ISOLATE HI CYL #
003054  033142  000006  000000          OR      R6,R2                           ; OR IN TO CALCULATED HI CYL

                                ; ***   CHECK IF SEEK NEEDED ***

003055  033441  000007  010000          MOV     R7,R1                           ; [U52EC2]MOVE LO CYL TO R1 - CYL NOW IN R2,R1
003056  010545  007012  135601          ADD     #SDI.CH,UBAR\N,BAR %CALL S.LDR7 ; R7=CURRENT CYL # HI
003057  010545  007026  135600          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDR6 ; R6=EXTENDED STATUS
003060  014546  000004  010000          BIT     #FSEEK,R6                       ; IF FORCE SEEK SET
003061  035546  013044  003073          BIC     #<FSEEK!URETRY>,R6,BUF %JNZRO U.IOPE ; [U52EC2]  THEN CLR FLAG AND DO SEEK
                                                                                ; [U52EC2]ALWAYS CLEAR URETRY FLAG
003062  010545  007011  125600          ADD     #SDI.CL,UBAR\N,BAR %CALL S.LDR6 ; R6=CURRENT CYL # LO
003063  036146  060001  103073          XOR     R1,R6           %JUPF   U.IOPE  ; IF REPLACE THEN SEEK                  ;[E121]
```



```
003064  036147  010002  013073          XOR     R2,R7           %JNZRO  U.IOPE  ; IF LO CYL'S DIFFER THEN SEEK            ;[E121]
                                                                                                                        ;[E121]
                                ;       The following code forces the lower processor to perform a group                ;[E121]
                                ;       select on this drive if the new request is on the current cylinder              ;[E121]
                                ;       but a different group. This gets the group select to the drive earlier          ;[E121]
                                ;       under two sets of circumstances:                                                ;[E121]
                                ;                                                                                       ;[E121]
                                ;       1)      In heavy multi-drive I/O where transfer activity on the other           ;[E121]
                                ;               drives can overlap with the group select.                               ;[E121]
                                ;       2)      When performing spiral reads on a single drive.                         ;[E121]
                                                                                                                        ;[E121]
003065  033746  010060  013073          MOV     #<SUSP+BFRQ>,R6 %JNZRO  U.IOPE  ; IF HI CYL'S DIFFER THEN SEEK          ;[E121]
003066  010545  007005  135604          ADD     #SDI.UG,UBAR\N,BAR %CALL S.LLBO ; GET CURRENT GROUP NUMBER              ;[E121]
003067  010550  007026  135610          ADD     #S.GRUP,R10\N,BAR %CALL S.XORO  ; COMPARE TO DESIRED GROUP              ;[E121]
003070  033746  010062  143100          MOV\F   #SUSP!BFRQ!SEEK,R6 %JZRO U.IOPF ; IF GROUPS EQUAL THEN SKIP IT         ;[E121]
003071  133546  000002  124664          BIS     #GSEL,R6        %CALL   U.SGRP  ; OTHERWISE TELL LOWER TO PERFORM       ;[E121]
003072  013440  000000  103100          NOP                     %JMP    U.IOPF  ; A GROUP SELECT NOW                    ;[E121]

                                ;       REVECTORING ENTERS HERE TO SEEK TO RCT CYLINDERS (PASSED IN R1,R2)
                                ;       UBAR --> SDI CTL BLK
                                ;       BAR --> SDI.CL(UBAR)
                                ;       Q = LEVEL 1 FRAME CODE FOR OPERATION
                                ;       R10 --> MSCP/PARAMETER BLOCK
                                ;       RETURN WITH FILLED-IN LEVEL 1 FRAME IN Q, SDI.ST IN RO, AND SEEK STARTED
                                ;       R7 IS PRESERVED BY THIS ROUTINE
                                ;       Q RETURNED WITH HEAD ADDRESS/COMMAND FROM S.TRAK

003073  033746  000062  124664  U.IOPE: MOV     #<SUSP!BFRQ!SEEK>,R6 %CALL U.SGRP; [U52EC1]R6=SDI STATUS BITS TO SET/RESET GROUP
003074  010545  007011  135620          ADD     #SDI.CL,UBAR\N,BAR %CALL S.STR1 ; SAVE LO CYL #
003075  010545  007012  125621          ADD     #SDI.CH,UBAR\N,BAR %CALL S.STR2 ; SAVE HI CYL #

                                ; ***   SEEK NEEDED - CLEAR SEEK FAIRNESS COUNTER ***

003076  010545  007026  125573          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDRO ; RO=FAIRNESS COUNTER
003077  135540  003177  000000          BIC     #FAIRCT,RO,BUF                  ; ZAP IT BECAUSE WE'RE SEEKING
003100  010550  007027  125573  U.IOPF: ADD     #S.TRAK,R10\N,BAR %CALL S.LDRO  ; RO=CURRENT TRACK NUMBER
003101  003040  003000  000000          OR      RO,Q,BUF                        ; STORE REAL TIME CMD/HEAD ADDR
003102  013440  000000  125476  U.SETS: NOP                     %CALL   U.GMST  ; GET MUTEXED SDI STATUS IN RO

                                ; ***   ENTER HERE WITH PLOCK CLEAR AND SDI.ST IN RO ***

003103  013740  007156  125574  U.SETX: MOV     #NSEEKS,BAR     %CALL   S.LDR1  ; [U52EC1]R1=CURRENT SEEK COUNT
003104  014546  000002  010000          BIT     #SEEK,R6                        ; IF SEEK NEEDED
003105  130461  010001  075620          INC\T   R1              %CNZRO  S.STR1  ; INCREMENT AND RESET
003106  010545  007000  000000          ADD     #SDI.ST,UBAR\N,BAR              ; BAR=PTR TO SDI STATUS                  ;[E121]
003107  033140  003006  115474  U.STSX: OR      R6,RO,BUF       %JMP    UNLOCK  ; SET NEW STATUS BITS/FREE D.PROC/RETURN

                                ; ***   ENTRY POINT #3 - TERMINATION PROCESSING ***
                                ; ***   ENTER WITH SDI STATUS IN RO, UPF SET IF REPLACE, R10 -> PKT, //R3 -> UNIT BLK
                                ; ***   FIRST RELEASE ALL BUFFERS ***

003110  033447  000000  125505  U.IOPG: MOV     RO,R7           %CALL   S.RELC  ; [16K]RETURN BUFR CNTRL BLKS,MOVE STATUS TO R7
003111  010550  007005  135600          ADD     #P.RSO3,R10\N,BAR %CALL S.LDR6  ; [U52EC2][ECO#1]R6=LO BYTE ORIG OP CODE
003112  010550  007007  135575          ADD     #P.MOD,R10\N,BAR %CALL  S.LDR2  ; R2=MSCP MODIFIER FIELD
```

```
003113  010550  007006  125604            ADD     #P.OPCD,R10\N,BAR %CALL S.LLBO  ; RO=MSCP OP CODE

                                ; ***   COMPARE COMMAND - CHECK FOR SECOND TIME AROUND ***

003114  016540  060040  113176            XOR     #OP.CMP,RO\N    %JUPF   U.IOPM  ; IF REPLACE THEN DONE/ELSE IF  COMPARE
003115  013446  010006  113144            TST     R6              %JZRO   U.IOPK  ;   THEN SKIP CHK FOR MODIFIER
003116  010545  007060  125600            ADD     #SDI.UF,UBAR\N,BAR %CALL S.LDR6 ;[V05] R6=SUBUNIT'S UNIT FLAGS          ;[E121]
003117  016540  000041  010000            XOR     #OP.RD,RO\N                     ; IF NOT READ COMMAND
003120  016560  010042  053121            XOR\T   #OP.WR,RO\N     %TNZRO          ; OR WRITE COMMAND

                                ; ***   READ OR WRITE COMMAND - CHECK FOR COMPARE MODIFIER OR COMPARE ON READ/WRITE UNIT FLAG ***

003121  114542  010100  003146            BIT     #MD.CMP,R2      %JNZRO  U.IOPL  ;   THEN DONE/IF COMPARE MODIFIER
003122                                    ASSUME  OP.RD,EQ,<OP.RD&OP.WR>+UF.CMR
003122                                    ASSUME  OP.WR,EQ,<OP.RD&OP.WR>+UF.CMW
003122  035540  010040  003125            BIC     #<OP.RD&OP.WR>,RO %JNZRO U.IOPI ;   THEN DO COMPARE/ELSE RO=UNIT FLAG BIT
003123  014146  000000  010000            BIT     RO,R6                           ; IF NO COMPARE UNIT FLAG
003124  013440  010000  103146            NOP                     %JZRO   U.IOPL  ;   THEN DONE

                                ; ***   SAVE ORIGINAL OP CODE, START AGAIN AS COMPARE ***

003125  010550  007006  125573  U.IOPI: ADD     #P.OPCD,R10\N,BAR %CALL S.LDRO  ; REFETCH OPCODE WITH END PKT FLAGS
003126  014547  000004  000000            BIT     #DERR,R7                        ; IF DRIVE ERROR
003127  013740  013040  053146            MOV\F   #OP.CMP,BUF     %JNZRO  U.IOPL  ;   THEN DONE/ELSE CONVERT TO COMPARE
003130  010550  007005  125616            ADD     #P.RSO3,R10\N,BAR %CALL S.STRO  ; SAVE OPCODE/FLAGS IN P.RSO3
003131  137746  000041  132375  U.IOPJ: COM     #<SLAT!DATT>,R6 %CALL   U.CLRS  ; ZAP ALL BUT SLAT AND DATT IN SDI.ST
003132  033746  000041  133102            MOV     #<SUSP+PKIP>,R6 %CALL   U.SETS  ; SET SUSPEND AND PKT IN PROGRESS
003133  033443  000605  122757            MOV     UBAR,R3 @RUPF   %CALL   U.IOCM  ;[V05] [16K]GO RECALCULATE AS IF NEW PKT
003134  033747  000022  132014            MOV     #<CMDCOF-BUF.SD>,R7 %CALL U.VAXP; SEE IF VAX PURGE NEEDED
                                                                                ; R7+BUF.SD=CONTAINS SDI CTRL BLK PTR FOR U.VAXP
003135  010545  007020  135630            ADD     #SDI.EO,UBAR\N,BAR %CALL S.CLRB ; [U52EC1]ZAP SDI ERROR STATE WORD

                                ;; BEGINNING OF RE-DO MAPPING INIT INCLUDING REFILL PTE CACHE
                                ;; FOR MAPPED COMMANDS AND NOT IN DM MODE

003136  010545  007115  135573            ADD     #MAP.ST,UBAR\N,BAR %CALL S.LDRO ; IS THIS A MAPPED COMMAND? [C118]
003137  114544  010100  113143            BIT     #DMODE,RLL      %JZRO   U.CMP   ; SKIP IF IN DM
003140  013440  010000  013143            NOP                     %JNZRO  U.CMP
003141  013440  000000  125232            NOP                     %CALL   U.MINT  ; RE-DO MAPPING INITIALIZATION
003142  033767  010026  052071            MOV\T   #ER.MRR,R7      %JNZRO  U.UERR  ;[RAE4-C119] if error say so
003143  033740  000040  102770  U.CMP:   MOV     #OP.CMP,RO      %JMP    U.IOPC  ; [16K]START UP THE I/O AGAIN

                                ; ***   CHECK FOR ORIGINAL COMPARE COMMAND ***

003144                                    ASSUME  OP.CMP,EQ,40                    ; [U52EC2]MAKE SURE
003144                                    ASSUME  OP.WR,EQ,42                     ; [U52EC2]FOLLOWING "OR"
003144                                    ASSUME  OP.RD,EQ,41                     ; [U52EC2]WILL WORK OK

003144  033506  010003  143146  U.IOPK: OR\F    (BUF),R6        %JZRO   U.IOPL  ; [U52EC2][ECO#1]IF P.RSO3 EQ 0 THEN CONTINUE
                                                                                ; [U52EC2]  ELSE OR OPCD WITH R6(RSO3)

003145  135546  003001  000000            BIC     #BIT08,R6,BUF                   ; [U52EC2][ECO#1]ZAP COMPARE FLAG AND STORE OPCODE

                                ; ***   CHECK FOR DATA SYNC TIMEOUT AND REPORT AS BAD BLOCK IF NOT IN RCT  ***
```



```
003146  010545  007003  135573  U.IOPL: ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDRO ; [U52EC2]RO=MSCP STATUS
003147  036540  000213  000000            XOR     #<ST.DRV!SC.RWR>,RO             ; [EERREC][U52EC2]IF NEQ DATA SYNC TIMEOUT CODE
003150  013440  010000  003153            NOP                     %JNZRO  U.IOPN  ; [U52EC2]  THEN CONTINUE
003151  013740  000001  134602            TST     #1              %CALL   U.GLBX  ; [U52EC2]  ELSE GET LBN IN R1,R2
003152  013440  000000  124604            NOP                     %CALL   U.SLBN  ; [U52EC2]STORE BAD BLOCK

                                ; ***   I/O COMPLETE - CALCULATE BYTES TRANSFERRED, ZAP PACKET RESERVED FIELDS ***

003153  010545  007115  125630  U.IOPN: ADD     #MAP.ST,UBAR\N,BAR %CALL S.CLRB ; done so clear mapping setup flag ;[E122]
003154  010550  007016  135573            ADD     #S.LBNL,R10\N,BAR %CALL S.LDRO  ; RO=FINAL LBN LO                       ;[E122]
003155  010550  007017  135600            ADD     #S.LBNH,R10\N,BAR %CALL S.LDR6  ; R6=FINAL LBN HI
003156                                    ASSUME  P.LBN1,EQ,P.LBNO+1              ; MAKE SURE SEQUENTIAL
003156  000550  007021  133211            ADD     #P.LBN1,R10,Q,BAR %CALL LDR2R1  ; R1/R2=STARTING LBN LO/HI
003157  131140  000001  000000            SUB     R1,RO                           ; RO=LO DIFFERENCE
003160  031146  020002  125611            SUBC    R2,R6           %CCRY   S.INC6  ; R6=HI DIFFERENCE
003161  033546  000100  010000            BIS     #100,R6                         ; SET TERMINATION FLAG FOR MULT BY 512
003162  030140  000000  010000  U.IOSH: ADD     RO,RO                           ; MULT LO BY 2
003163  030146  020006  125611            ADD     R6,R6           %CCRY   S.INC6  ; MULT HI BY 2
003164  050146  030006  013162            ADD\R   R6,R6           %JNMSB  U.IOSH  ; IF NO FLAG THEN CONTINUE
003165                                    ASSUME  P.BCN1,EQ,P.BCNO+1              ; MAKE SURE SEQUENTIAL
003165  000550  007011  133211            ADD     #P.BCN1,R10,Q,BAR %CALL LDR2R1  ; R1/R2=ORIGINAL BYTE CNT LO/HI
003166  112146  000002  010000            CMP     R2,R6                           ; COMPARE HI BYTE COUNTS
003167  112160  010001  153170            CMP\T   R1,RO           %TZRO           ; IF EQ THEN COMPARE LO BYTE COUNTS
003170  033441  020000  053172            MOV\F   RO,R1           %JNCRY  U.IOSA  ; IF ORIGINAL LEQ CALCULATED THEN DONE
003171  033442  000006  000000            MOV     R6,R2                           ;   ELSE SWAP BYTE COUNTS
003172  013440  000000  123210  U.IOSA: NOP                     %CALL   U.SBCN  ; GO SET BYTE COUNT

                                ; ***   SET IN BAD BLOCKS - IF ANY ***

003173                                    ASSUME  SDI.BH,EQ,SDI.BL+1              ; [U52EC1]MAKE SURE SEQUENTIAL
003173  000545  007030  123211            ADD     #SDI.BH,UBAR,Q,BAR %CALL LDR2R1 ; [U52EC1]R1/R2=SAVED 1ST BAD BLOCK

                                ; ***   NOW ZAP BAD BLOCK STORAGE FOR NEXT COMMAND ***
                                ; ***   Q AND BAR POINT TO SDI.BL

003174  033740  003000  123370            MOV     #0,RO,BUF       %CALL   STROLG  ; [U52EC1]ZAP RO & SDI.BL THEN INCR Q,ZAP SDI.BH
003175                                    ASSUME  P.FBB1,EQ,P.FBB0+1              ; MAKE SURE SEQUENTIAL
003175  000550  007020  133213            ADD     #P.FBB0,R10,Q,BAR %CALL STR1R2  ; STORE IN CORRECT PLACE

                                ; ***   CALCULATE BYTES TRANSFERRED, ZAP PACKET RESERVED FIELDS ***
                                ; ***   IF REPLACE THEN ZAP END OF PACKET ***

003176  033441  000010  010000  U.IOPM: MOV     R10,R1                          ; R1=MSCP PKT PTR
003177  030541  000012  010000            ADD     #P.BUFO,R1                      ; R1 POINTS AT START OF BUFFER DESCRIPTOR
003200  033740  000006  135544            MOV     #<P.BUF5+1>-P.BUFO,RO %CALL S.ZBUF; ZAP BUFFER DESCRIPTOR
003201  033741  060010  123206            MOV     #P.RSO6,R1      %CUPF   U.PKTZ  ; IF NOT REPLACE THEN CONTINUE
003202  010550  007005  135630            ADD     #P.RSO3,R10\N,BAR %CALL S.CLRB  ; ZAP COMPARE WORK WORD

                                ; ***   OBTAIN STATUS CODE AND SUBCODE FROM SDI.SW **

003203  013440  000000  133640            NOP                     %CALL   U.DONX  ; GO SEND END PACKET WITH CODE FROM SDI.SW
003204  010545  007000  125574            ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR1 ; [U52EC1]R1=SDI STATUS
003205  014541  000001  103766            BIT     #PKIP,R1        %JMP    U.CSDI  ; [16K][U52EC1]GO TRY TO START NEXT PKT
```

```
                                ; ***   ZAP END PACKET FOR REPLACE,AVAILABLE,DETERMINE ACCESS PATHS COMMANDS ***

003206  033740  000022  010000  U.PKTZ: MOV     #<MCP.LN+P-P.RSO6>,R0           ; R0=WORDS TO ZAP FOR REPL/AVAIL/DET.ACC.PATH
003207  030141  000010  105544          ADD     R10,R1          %JMP    S.ZBUF  ; GO ZAP BUFFER/RETURN

003210                          U.SBCN: ASSUME  P.BCN1,EQ,P.BCN0+1              ; MAKE SURE SEQUENTIAL
003210  000550  007010  103213          ADD     #P.BCN0,R10,Q,BAR %JMP  STR1R2  ; STORE LO/HI BYTE COUNT/RETURN

                                ; ***   SUBROUTINE TO LOAD R2,R1 FROM BAR, AND Q-1 ***

003211  033702  000003  010000  LDR2R1: MOV     (BUF),R2                        ; R2=HI WORD
003212  101640  007001  115574          SUB     #1,Q,BAR        %JMP    S.LDR1  ; [U52EC2]BAR,Q=Q-1/R1=LO WORD/RETURN

                                ; ***   SUBROUTINE TO STORE R1,R2 IN BAR AND Q+1 ***

003213  013440  003001  010000  STR1R2: MOV     R1,BUF                          ; BUF=LO WORD
003214  000640  007001  115621          ADD     #1,Q,BAR        %JMP    S.STR2  ; [16K]Q,BAR=Q+1/STORE HI WORD/RETURN

                                ; ***   SUBROUTINE TO COMPARE LBN(R1,R2) WITH START RCT LBN ***

003215  010545  007075  135600  U.CRCT: ADD     #SDI.L1,UBAR\N,BAR %CALL S.LDR6 ;[V05] [16K]R6=LO START RCT LBN
003216  010545  007076  125601          ADD     #SDI.L2,UBAR\N,BAR %CALL S.LDR7 ;[V05] [16K]R7=HI START RCT LBN
003217  112147  000002  000000          CMP     R2,R7                           ; [16K]IF HI LBN EQ HI RCT LBN
003220  112166  010001  143221          CMP\T   R1,R6           %TZRO           ; [16K]  THEN COMPARE LO LBN WITH LO RCT LBN/RETURN
003221  013440  000000  177777          NOP\F                   %RET            ; [16K]RETURN WITH CCODES INTACT
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.LNKP (LINK MSCP PACKET TO QUEUE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL LINK A MSCP PACKET TO A QUEUE WHOSE HEAD IS
                                ; POINTED TO BY BAR.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET TO BE QUEUED
                                ;       BAR             POINTER TO MSCP PACKET QUEUE HEAD
                                ;
                                ; OUTPUTS:
                                ;       R1 IS USED AS A TEMPORARY REGISTER
                                ;       PACKET ENTERED ON QUEUE
                                ;
                                ;;-
003222  033701  000003  113224  U.LNKP: MOV     (BUF),R1        %JMP    U.LNKB  ; R1=MSCP PKT QUEUE WORD, ENTER LOOP
003223  013440  007001  125574  U.LNKA: MOV     R1,BAR          %CALL   S.LDR1  ; R1=PKT QUEUE WORD
003224  115541  000300  010000  U.LNKB: BIC     #Q.STAT,R1\N                    ; ISOLATE NEXT LINK PTR
003225  033150  013001  053223          BIS\F   R1,R10,BUF      %JNZRO  U.LNKA  ; IF NEQ 0 THEN LOOK AT NEXT
003226  013440  007010  000000          MOV     R10,BAR                         ; BAR=PKT STATUS WORD PTR
003227  113740  003200  127777          MOV     #PSTAT,BUF      %RET            ; SET PKT IN USE/RETURN

                                ; ***   ROUTINE TO LINK R10 TO HEAD OF QUEUE ***

003230  010545  007022  135575  U.LNKH: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LDR2 ; NOW R2=1ST ENTRY ON LIST
003231  133542  000300  135626          BIS     #<PSTAT+PACTV>,R2 %CALL S.ST10  ; SET NEW LIST HEAD, SET FLAGS IN
003232  013440  007010  115621          MOV     R10,BAR         %JMP    S.STR2  ; [U52EC1]OLD LIST HEAD AND MAKE NEW NEXT PTR
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.LOGS (CONTROLLER LOG AND ATTENTION MESSAGE HANDLING ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PROCESS REQUESTS TO SEND BOTH ERROR LOG AND
                                ; ALL ATTENTION MESSAGES, I.E. 'SLAT' BIT SET IN SDI CONTROL
                                ; BLOCK STATUS WORD. IF THE LOG/ATTENTION PACKET (LOGPKT) IS NOT FREE
                                ; THEN AN IMMEDIATE RETURN IS MADE.
                                ;
                                ; FOR ATTENTIONS IT WILL DO THE FOLLOWING:
                                ;       1. IF ATTENTIONS NOT ENABLED BY HOST THEN CLEAR SLAT AND RETURN
                                ;       2. IF ACCESS PATH OR DUPLICATE ATTENTION THEN PUT UNIT NUMBER IN LOG
                                ;          PACKET, CLEAR SLAT AND RETURN.
                                ;       3.    ELSE SEARCH THE SUBUNIT CHARACTERISTICS BLOCKS FOR A MATCH
                                ;             ON UBAR
                                ;       4. IF NO MATCH FOUND THEN CLEAR 'SLAT' AND RETURN
                                ;       4.    ELSE CLEAR 'DRV.AT' IN SDI.CW AND BUILD ATTENTION MESSAGE IN 'LOGPKT'
                                ;       5. QUEUE ATTENTION PACKET ON CONTROLLER -> HOST QUEUE AND RETURN
                                ;
                                ; FOR ERROR LOGS IT WILL DO THE FOLLOWING:
                                ;       1. DECODE ERROR LOG FORMAT
                                ;       2. BUILD APPROPRIATE ERROR LOG PACKET IN 'LOGPKT'
                                ;       3. QUEUE ERROR LOG PACKET ON CONTROLLER -> HOST QUEUE
                                ;       4. CLEAR 'SLAT' AND RETURN
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       ERROR LOG OR ATTENTION MESSAGE QUEUED TO 'HSTQUE'
                                ;
                                ;;-

                                ; ***   CHECK TO SEE IF LOG PACKET AVAILABLE ***

003233  033750  007304  135573  U.LOGS: MOV     #LOGPKT,R10,BAR %CALL   S.LDRO  ; RO=STATUS WORD FOR LOG/ATTN PKT
003234  033740  010036  037777          MOV     #LOG.LN+P,RO    %RNZRO          ; IF IN USE THEN RETURN/ELSE RO=LENGTH TO CLEAR
003235  033441  007010  125544          MOV     R10,R1,BAR      %CALL   S.ZBUF  ; GO ZAP ENTIRE BUFFER
003236  013740  007170  135572          MOV     #CNTFLG,BAR     %CALL   S.LDQQ  ; Q=CONTROLLER FLAGS
003237  010545  007003  125574          ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDR1 ; R1=EVENT CODE
003240  010550  007007  135620          ADD     #L.EVNT,R10\N,BAR %CALL S.STR1  ; STORE EVENT CODE IN PKT
003241  010545  007010  125573          ADD     #SDI.1T,UBAR\N,BAR %CALL S.LDRO ; RO=LOG/ATTN CODE
003242  010550  007006  010000          ADD     #L.FMT,R10\N,BAR                ; BAR=PTR TO LOG FORMAT
003243  014540  000100  010000          BIT     #ATTCOD,RO                      ; IF NOT ATTENTION
003244  014640  010200  113262          BIT     #CF.ATN,Q       %JZRO   ER.LOG  ;   THEN DO ERROR LOG CHECK
003245  133746  010040  112375  U.LOGA: MOV     #SLAT,R6        %JZRO   U.CLRS  ; [ECO#2]IF ATTN'S NOT ENABLED THEN CLEAR SLAT

                                ; ***   ATTENTION MESSAGE PROCESSING ***

003246  033742  000014  135616          MOV     #DUP.LN,R2      %CALL   S.STRO  ; R2=DUPL/ACC PATH LENGTH/SAVE ATTN CODE
003247  010545  007043  135574          ADD     #SDI.UN,UBAR\N,BAR %CALL S.LDR1 ; R1=UNIT NUMBER
003250  135541  000374  000000          BIC     #DRV.UM,R1                      ; ISOLATE 10 BITS OF IT
003251  010550  007004  135620          ADD     #P.UNIT,R10\N,BAR %CALL S.STR1  ; STORE LO BYTE IN PACKET

KDBUP   KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```



```
003252  016540  000100  000000          XOR     #OP.AVA,RO\N                    ; IF NOT AVAIABLE ATTENTION MSG
003253  013440  010000  013336          NOP                     %JNZRO  U.CSLA  ; [UQA][16K]THEN GO CLEAR SLAT & QUE MSG

                                ;       ASSUME  UN.BUF&1,EQ,0                   ; [16K]MAKE SURE UN.BUF IS EVEN
                                ; ***   ATTENTION MESSAGE - CHECK FOR MULTI-UNITS ***

003254  010545  007056  135573          ADD     #SDI.CW,UBAR\N,BAR %CALL S.LDRO ;[V05] RO=SUBUNIT BLK STATUS
003255  114540  010004  102375          BIT     #DRV.AT,RO      %JZRO   U.CLRS  ; IF NOT ACTIVE THEN CONTINUE
003256  135540  013004  152375          BIC\F   #DRV.AT,RO,BUF  %JZRO   U.CLRS  ; IF NOT SENT THEN GO DO IT

                                ; ***   NEED TO SEND ATTENTION FOR THIS UNIT ***

003257  010550  007004  000000          ADD     #P.UNIT,R10\N,BAR               ; BAR=PTR TO PKT UNIT NUMBER
003260  135540  003374  133702          BIC     #DRV.UM,RO,BUF  %CALL   U.SUNI  ; GO FILL IN ATTENTION PACKET
003261  033742  000040  113337          MOV     #AVL.LN,R2      %JMP    U.SLEN  ; GO SET LENGTH/QUE PKT TO HOST

                                ; ***   ERROR LOG PROCESSING ***

003262  013441  000001  000000  ER.LOG: TST     R1                              ; [ECO#2]IF SDI.SW = 0
003263  114540  010300  143245          BIT\F   #<LF.CON!LF.SUC>,RO %JZRO U.LOGA; [ECO#2] THEN DONE/IF CONTINUING OR SUCCESS
003264  113540  013001  033371          BIS     #LF.SNR,RO\N,BUF %CNZRO U.ZPSW  ;   THEN SET SEQ NUMBER RESET/ZAP SDI.SW
003265  010545  007022  125600          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LDR6 ; R6=MSCP PKT PTR
003266  010545  007006  135601          ADD     #SDI.UB,UBAR\N,BAR %CALL S.LDR7 ; R7=DATA BUFFER CONTROL PTR
003267  010550  007001  000000          ADD     #P.VCID,R10\N,BAR               ; BAR=VIRTUAL CIRCUIT ID PTR
003270  013740  003020  135605          MOV     #VC.LOG,BUF     %CALL   S.MLBO  ; SET TO ERROR LOG ID
003271  016540  000000  010000          XOR     #FM.CNT,RO\N                    ; IF CONTROLLER LOG (LAST FAILURE)
003272  016540  010001  113304          XOR     #FM.BAD,RO\N    %JZRO   LG.CNT  ;   THEN GO DO IT
003273  016540  010002  103341          XOR     #FM.DSK,RO\N    %JZRO   LG.BAD  ; IF BUS ADDR LOG THEN GO DO IT
003274  000550  010027  103346          ADD     #<L.SDIO-1>,R10,Q %JZRO LG.DSK  ; IF NO MATCH THEN CLEAR SLAT/RETURN

                                ; ***   ERROR LOG MESSAGE FORMAT 3 - SDI ERROR ***

003275  010545  007044  123365  LG.SDI: ADD     #SDI.S1,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 1
003276  010545  007045  133365          ADD     #SDI.S2,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 2
003277  010545  007050  123365          ADD     #SDI.S4,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 4
003300  010545  007051  133365          ADD     #SDI.S5,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 5
003301  010545  007052  133365          ADD     #SDI.S6,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 6
003302  010545  007053  123365          ADD     #SDI.S7,UBAR\N,BAR %CALL STBFLG ; RO=SDI STATUS WORD 7
003303  033742  000070  103314          MOV     #SDI.LN,R2      %JMP    ER.UID  ; GO SET UNIT ID,FINISH UP PKT/EXIT

                                ; ***   ERROR LOG MESSAGE FORMAT 0 - CONTROLLER ERROR ***

003304  013740  007034  135573  LG.CNT: MOV     #FAILUR,BAR     %CALL   S.LDRO  ; RO=LAST FAILURE CODE
003305  112540  000143  010000          CMP     #99.,RO                         ; [mjt07] make sure that we don't
                                ;                                                clear the bda specific values
003306  112540  030150  013310          CMP     #BERMAX,RO      %JNNEG  1$      ; [mjt07] if <, go check codes < 37(q)
003307  013440  030000  003312          NOP                     %JNNEG  2$      ; [mjt07] if ok, go store it
003310  015540  000037  000000  1$:     BIC     #37,RO\N                        ; IF LAST FAIL CODE NOT VALID
003311  034460  010000  053312          CLR\T   RO              %TNZRO          ;   THEN RO=0
003312  010550  007015  135616  2$:     ADD     #L.MLUN,R10\N,BAR %CALL S.STRO  ; STORE IN PKT
003313  033742  000030  103334          MOV     #CNT.LN,R2      %JMP    ER.CID  ; R2=CONTROLLER LOG LENGTH

                                ; ***   COMMON EXIT POINT TO STORE UNIT ID, VOL SERIAL NUMBER, UNIT# ***

KDBUP   KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```

```
003314  000550  000015  133675  ER.UID: ADD     #<L.UNTO-1>,R10,Q %CALL U.SUID  ; POINT TO COMMON LOG PKT AREA TO STORE
003315  010545  007036  135573          ADD     #SDI.RS,UBAR\N,BAR %CALL S.LDRO ; RO=H/W & S/W VERSION NUMBER   ;[E121]
003316  135540  000200  133370          BIC     #FDIAG,RO          %CALL STROLG ; but clear the FD bit          ;[E121]
003317  000640  000001  010000          ADD     #1,Q                            ; [ECO#1]SKIP OVER RETRY
003320  010545  007061  133365          ADD     #SDI.V1,UBAR\N,BAR %CALL STBFLG ;[V05] RO=VOL SERIAL NUMBER #1
003321  010545  007062  133365          ADD     #SDI.V2,UBAR\N,BAR %CALL STBFLG ;[V05] RO=VOL SERIAL NUMBER #2
                        ;                 ADD     #SDI.CW,UBAR\N,BAR %CALL S.LDRO ;[V05] RO=UNIT NUMBER
003322  010545  007043  125573          ADD     #SDI.UN,UBAR\N,BAR %CALL S.LDRO ;[V05] RO=UNIT NUMBER    [rae06]
                        ;                 BIC     #DRV.UM,RO                      ; ISOLATE UNIT NUMBER
003323  135540  000360  010000          BIC     #DRV.SU,RO                      ; ISOLATE UNIT NUMBER    [rae06]
003324  010550  007004  135616          ADD     #L.UNIT,R10\N,BAR %CALL S.STRO  ; STORE IN LOG PKT
003325  000550  000014  133401          ADD     #<L.MLUN-1>,R10,Q %CALL U.MLUN  ; GO CREATE AND STORE MULTI-UNIT CODE

                        ; ***   SET REFERENCE NUMBER, OTHER MSCP THINGS ***

003326                  ER.SRN: ASSUME  <PKTO20&BITOO>,EQ,0             ; [16K]MAKE SURE CHECK IS VALID
003326                          ASSUME  <<SDI.1!SDI.2!SDI.3!SDI.4>&BITOO>,NE,0; [16K]DITTO
003326                          ASSUME  <SDI.AT&BITOO>,EQ,0             ; [16K]DITTO
003326  014546  000001  010000          BIT     #BITOO,R6               ; [16K]IF PKT IN SDI CNTRL BLK AREA
003327  000550  010001  003334          ADD     #<L.CRFO-1>,R10,Q %JNZRO ER.CID ;   THEN SKIP/ELSE POINT TO LOG REFERENCE NUMBER
003330  010546  007002  133365          ADD     #P.CRFO,R6\N,BAR %CALL  STBFLG  ; STORE LO REF # IN LOG PKT
003331  010546  007003  123365          ADD     #P.CRF1,R6\N,BAR %CALL  STBFLG  ; STORE HI REF # IN LOG PKT
003332  010546  007006  135573          ADD     #P.FLGS,R6\N,BAR %CALL  S.LDRO  ; RO=MSCP FLAGS
003333  133540  003040  000000          BIS     #EF.LOG,RO,BUF                  ; INDICATE ERROR LOG GENERATED

                        ; ***   ERROR LOG FILL IN CONTROLLER ID, H/W AND S/W VERSIONS ***

003334  000550  000007  137751  ER.CID: ADD     #<L.CNTO-1>,R10,Q %CALL U.SCID  ; GO SET CONTROLLER ID
003335                          ASSUME  <L.CSVR>,EQ,<L.CNT3+1>          ; MAKE SURE Q IS KOSHER
003335  033740  000023  127750          MOV     #CODVER,RO      %CALL   U.HWVR  ; RO=CONTROLLER SW VERSION NUMBER /
                                                                        ;  SET IN HW VERSION NUMBER.

                        ; ***   EXIT TO CLEAR SLAT FLAG IN SDI.ST AND QUE PACKET TO HOST ***

003336  133746  000040  132375  U.CSLA: MOV     #SLAT,R6        %CALL   U.CLRS  ; ZAP SLAT

                        ; ***   EXIT TO SET LOGLEN EQUAL TO R2 AND QUEUE PACKET TO HOST

003337  013740  007154  135621  U.SLEN: MOV     #LOGLEN,BAR     %CALL   S.STR2  ; SAVE PKT LENGTH (IN R2)

                        ; ***   ENTRY TO QUEUE A PACKET TO SEND TO THE HOST ***

003340  013740  007204  113222  U.QSND: MOV     #HSTQUE,BAR     %JMP    U.LNKP  ; GO LINK TO CONTROLLER->HOST QUEUE/RETURN

                        ; ***   ERROR LOG MESSAGE FORMAT 1 - UNIBUS ACCESS ERROR ***

003341  000550  000015  000000  LG.BAD: ADD     #<L.BADO-1>,R10,Q               ; Q=START OF STORE AREA-1
003342  010547  007007  123365          ADD     #BUF.UA,R7\N,BAR %CALL  STBFLG  ; GET & SAVE LO UNIBUS ADDRESS
003343  010547  007010  135573          ADD     #BUF.US,R7\N,BAR %CALL  S.LDRO  ; RO=HI UNIBUS ADDRESS
003344  135540  000300  123370          BIC     #BANOT,RO       %CALL   STROLG  ; [BDA] STORE IN LOG PKT
003345  033742  000034  113326          MOV     #BAD.LN,R2      %JMP    ER.SRN  ; GO FILL IN REST OF LOG PKT

                        ; ***   ERROR LOG MESSAGE FORMAT 2 - DISK TRANSFER ERROR ***
```



```
003346  000550  000022  010000  LG.DSK: ADD     #<L.RTRY-1>,R10,Q               ; [ECO#1]SET UP Q FOR RETRY FIELD
003347  010545  007017  123360          ADD     #SDI.E1,UBAR\N,BAR %CALL STBFLH ;[V05] GET AND SAVE RETRY COUNT
003350  133740  000200  010000          MOV     #LF.SUC,RO                      ; if this is a successful retry ;[E121]
003351  010550  007006  135603          ADD     #L.FMT,R10\N,BAR    %CALL S.BITO ;   then                        ;[E121]
003352  013440  010000  113354          NOP                         %JZRO 5$    ;                               ;[E121]
003353  010545  007017  125630          ADD     #SDI.E1,UBAR\N,BAR %CALL S.CLRB ;       CLEAR the retry state   ;[E121]
                        5$:                                                                                     ;[E121]
003354  000550  000025  000000          ADD     #<L.HDRO-1>,R10,Q               ; [ECO#2]SET UP Q FOR STORE
003355  010545  007031  133365          ADD     #SDI.RL,UBAR\N,BAR %CALL STBFLG ; [ECO#2][ECO#1]GET LO HDR FROM SDI.RL
003356  010545  007032  133365          ADD     #SDI.RH,UBAR\N,BAR %CALL STBFLG ; [ECO#2][ECO#1]GET HI HDR FROM SDI.RH
003357  033742  000054  103314          MOV     #DSK.LN,R2      %JMP    ER.UID  ; R2=DISK LOG LENGTH

                        ; ***   ROUTINES TO STORE INFORMATION SEQUENTIALLY IN ARRAY ACCESSED BY Q ***

003360  033700  000003  000000  STBFLH: MOV     (BUF),RO                        ;[V05]STORE (BUF) IN RO
003361  135540  010200  113370          BIC     #BIT15,RO       %JZRO   STROLG  ;[mjt08][V05]CLEAR DERR AND STORE IN ARRAY
003362  014540  000377  000000          BIT     #LOBYT,RO                       ;[mjt08]check level of retries
003363  031440  010000  143364          DEC\F   RO                      %TZRO   ;[mjt08]decr if not zero to reflect the proper sdi recovery
003364  013440  000000  103370          NOP                             %JMP    STROLG  ;[mjt08] then store it.

003365  033700  000003  103370  STBFLG: MOV     (BUF),RO        %JMP    STROLG  ; ENTER HERE TO STORE (BUF) IN ARRAY
003366  033700  000003  000000  STBBLG: MOV     (BUF),RO                        ; ENTER HERE TO STORE LO BYTE OF (BUF) IN ARRAY
003367  034540  000377  010000          AND     #LOBYT,RO                       ; ISOLATE LO BYTE
003370  000640  007001  105616  STROLG: ADD     #1,Q,BAR        %JMP    S.STRO  ; ENTER HERE TO STORE RO IN ARRAY

003371  010545  007003  115630  U.ZPSW: ADD     #SDI.SW,UBAR\N,BAR %JMP S.CLRB  ; ZAP STATUS WORD/RETURN
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.MIOP (MAINTENANCE I/O PEPARATION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PREPARE THE NECESSARY I/O PARAMETERS NEEDED
                                ; TO PERFORM MAINTENANCE I/O.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP MAINTENANCE I/O PACKET
                                ;
                                ; OUTPUTS:
                                ;       R0              TRANSFER BYTE COUNT (WILL BE EVEN)
                                ;       IUAR            UNIBUS ADDRESS
                                ;       UBAR            CONTROLLER RAM ADDRESS
                                ;
                                ;;-

003372  010550  007022  135572  U.MIOP: ADD     #P.RG00,R10\N,BAR %CALL S.LDQQ  ; Q=RELATIVE POSITION IN MEMORY
003373  000640  000012  135534  U.MIOA: ADD     #DMODT,Q        %CALL   S.FUNB  ; [16K]UBAR=DM.BEG ADDRESS
003374  030045  000005  000000          ADD     Q,UBAR                          ; [16K]UBAR=DM STARTING ADDRESS
003375  010550  007012  135600  U.MIOB: ADD     #P.BUFL,R10\N,BAR %CALL S.LDIU  ; [16K]IUAR=LO ORDER BUFFER DESCRIPTOR
003376  010550  007013  125536          ADD     #P.BUFH,R10\N,BAR %CALL S.SXB1  ; R1=HI ORDER BUFFER DESCRIPTOR
003377  010550  007010  135573          ADD     #P.BCNO,R10\N,BAR %CALL S.LDRO  ; R0=# BYTES TO TRANSFER
003400  055540  000001  137777          BIC\R   #1,R0           %RET            ; GET # WORDS TO TRANSFER IN R0
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.MLUN  (CREATE AND STORE MSCP MULTI-UNIT CODE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE CREATES AND STORES THE MSCP MULTI-UNIT CODE
                                ; IN THE AREA POINTED TO BY Q.
                                ;
                                ; INPUTS:
                                ;       Q               AREA WHERE MULTI-UNIT CODE IS TO BE STORED
                                ;       UBAR            SDI CONTROL BLOCK POINTER
                                ;
                                ; OUTPUTS:
                                ;       MULTI-UNIT CODE STORED IN AREA POINTED TO BY Q
                                ;
                                ; *** NOTE STACK AT MAXIMUM DEPTH ***
                                ;
                                ;;-

                                ; ***   CREATE AND FILL IN MULTI-UNIT CODE ***

003401                                  ASSUME  SDI.1,EQ,1355                   ; [U52EC1]MAKE SURE SCHEME FOR
003401                                  ASSUME  SDI.2,EQ,1475                   ; [U52EC1]GETTING 0,1,2,3 AS
003401                                  ASSUME  SDI.3,EQ,1615                   ; [U52EC1]MULTI-UNIT CODES
003401                                  ASSUME  SDI.4,EQ,1735                   ; [U52EC1]WILL WORK
003401  053440  000005  010000  U.MLUN: MOV\R   UBAR,R0                         ; [16K]R0=UBAR/2
003402  053440  000000  010000          ROT\R   R0,R0                           ; [16K]R0=UBAR/4
003403  053440  000000  010000          ROT\R   R0,R0                           ; [V05]R0=UBAR/8
003404  030540  000004  010000          ADD     #4,R0                           ; [rae01]ADJUST TO MAKE 0,1,2,3
                                                                                ;                    not 2,3,0,1
003405  054540  000006  113370          AND\R   #6,R0           %JMP    STROLG  ; [16K]R0=SDI CONTROL BLOCK PTR,STORE & RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.MNRD (MSCP MAINTENANCE READ COMMAND)
                                ;       'SEQUENTIAL COMMAND'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'MAINTENANCE READ' COMMAND.
                                ; THE FUNCTION PERFORMED IS DEPENDENT UPON THE VALUE OF REGION ID AND
                                ; ARE DESCRIBED BELOW:
                                ;       REGION ID = 1
                                ;               TRANSFERS DATA FROM THE CONTROLLER BUFFER MEMORY TO THE HOST
                                ;               BYTE COUNT AND REGION OFFSET DEFINE THE REGION OF BUFFER
                                ;               MEMORY TO BE TRANSFERED TO THE HOST.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;;-

003406  036542  000001  000000  U.MNRD: XOR     #1,R2
003407  033760  010001  042420          MOV\T   #ST.CMD,R0      %JNZRO  U.C2EP  ; IF REG ID NEQ 1 THEN ERROR EXIT

                                ; ***   MAINTENANCE READ CONTROLLER RAM READ FUNCTION ***

003410  013440  000000  123372          NOP                     %CALL   U.MIOP  ; [16K]GO CALCULATE I/O PARAMS
003411  013440  010000  036242          NOP                     %CNZRO  UNB.WR  ; WRITE BUFFER TO HOST
003412  013440  000000  113420          TST     R0              %JMP    U.MWDA  ; TEST R0/CONTINUE
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.MNWR (MSCP MAINTENANCE WRITE COMMAND)
                                ;       'SEQUENTIAL COMMAND'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'MAINTENANCE WRITE' COMMAND.
                                ; THE REGION ID AND BYTE ARE DECODED TO PERFORM THE FOLLOWING FUNCTIONS:
                                ;       REGION ID = 1 AND BYTE COUNT > 0
                                ;               DOWNLINE LOADS FROM HOST MEMORY INTO CONTROLLER BUFFER STARTING
                                ;               AT THE REGION OFFSET. BYTE COUNT MUST BE EVEN.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP PACKET
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;;-

003413  033760  010001  042420  U.MNWR: MOV\T   #ST.CMD,R0      %JNZRO  U.C2EP  ; IF REGION NEQ 1 THEN ERROR EXIT
003414  013440  000000  123372          NOP                     %CALL   U.MIOP  ; [16K]GO PREPARE FOR MAINT I/O
                                ;;;;    BIS\F   #<DMODE>,RLL    %JZRO   U.MWDB  ; if 0 leng jump                        [rae3-c119]
003415  133544  010120  143421          BIS\F   #<DMODE!DXFC>,RLL %JZRO U.MWDB  ; if 0 leng jump                        [rae3-c119]
003416  114544  010020  003416  1$:     BIT     #DXFC,RLL       %JNZRO  1$      ;  else wait for DPROC to clr DXFC       [rae3-c119]
                                                                                ;  indicating that it is now really     [rae3-c119]
                                                                                ;  in DMODE                              [rae3-c119]
003417  013440  000000  125635          NOP                     %CALL   UNB.RD  ; xfer data                             [rae3-c119]
003420  033760  010011  052420  U.MWDA: MOV\T   #ST.HST,R0       %JNZRO  U.C2EP  ; R0=HOST ACCESS ERROR/CHG TO END PKT
003421  033740  000000  112420  U.MWDB: MOV     #ST.SUC,R0      %JMP    U.C2EP  ; R0=SUCCESS/GO CHG TO END PKT
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.ONLN (MSCP ONLINE COMMAND)
                                ;       U.SUCH (MSCP SET UNIT CHARACTERISTICS COMMAND)
                                ;       'SEQUENTIAL COMMANDS'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP ONLINE COMMAND AND THE MSCP
                                ; SET UNIT CHARACTERISTICS COMMAND. INITIAL ENTRY WILL QUEUE THE MSCP
                                ; PACKET TO THE CORRECT SDI CONTROL BLOCK. THE SECOND
                                ; ENTRY (WHEN THE PACKET IS ACTIVE) WILL INITIATE THE FUNCTION INVOLVING
                                ; COORDINATION WITH THE DRIVE PROCESSOR.
                                ;       THE SET UNIT CHARACTERISTICS COMMAND IS FILTERED OUT AND BYPASSES
                                ; THE ONLINE SPECIAL PROCESSING.
                                ;
                                ; INPUTS:
                                ;       R0              MSCP OP CODE
                                ;       R1              MSCP PACKET STATUS AND QUEUE WORD
                                ;       R10             POINTER TO MSCP PACKET (1ST, 2ND, AND 3RD ENTRY)
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (2ND AND 3RD ENTRY)
                                ;
                                ; OUTPUTS:
                                ;       MSCP ONLINE 'END' PACKET
                                ;       MSCP SET UNIT CHARACTERISTICS 'END' PACKET
                                ;
                                ;;-

020000                          M.SUCH  := MD.CSE                                       ; modifiers allowed for OP.SUC          [rae02]
000044                          M.SUCL  := MD.EXC!MD.SWP                                ; [ch10] modifiers allowed for OP.SUC[rae02]
                                U.SUCH:
003422  103740  000040  010000          MOV     #M.SUCH,Q                               ; set modifiers                         [rae02]
003423  003640  000044  113426          BIS     #M.SUCL,Q       %JMP    U.ONLO  ; set legal modifier bits               [rae02]

020000                          M.ONLH: := MD.CSE                                       ; modifiers for OP.ONL                  [rae02]
000047                          M.ONLL: := MD.EXC!MD.SWP!MD.IMF!MD.RIP                  ; modifiers for OP.ONL          [ch10][rae02]
003424  103740  000040  010000  U.ONLN: MOV     #M.ONLH,Q                               ; set legal modifier bits               [rae02]
003425  003640  000047  000000          BIS     #M.ONLL,Q                               ; set legal modifier bits               [rae02]

                                U.ONLO:                                                                                         ;[rae02]
003426  114541  000100  010000          BIT     #PACTV,R1                               ; IF PACKET ACTIVE
003427  013440  010000  003441          NOP                             %JNZRO  U.ONLB  ;   THEN ENTRY POINT #2

                                ; ***   ENTRY POINT #1 - INITIAL PACKET PROCESSING ***

003430  013440  000000  122263          NOP                             %CALL   TSTMOD  ; test for reserved modifier bits       [rae02]
003431  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error                 [rae02]
003432  013440  000000  122256          NOP                             %CALL   MLN44   ; test that msg leng is large enough    [rae02]
003433  034441  010001  002506          CLR     R1                      %JNZRO  U.CMDA  ;   if not declare error               [rae02]
003434  013440  000000  122241          NOP                             %CALL   U.RSVA  ;  test reserved fields                 [rae02]
003435  013440  010000  002506          NOP                             %JNZRO  U.CMDA  ;   if err call err rtn                [rae02]
                                                                                        ; U.ONLA called by U.AVAL, so do test   [rae02]
                                                                                        ; of resv. fields first.                [rae02]
003436  013440  000000  122551  U.ONLA: NOP                             %CALL   U.GSDI  ; GO FIND SDI CONTROL BLOCK/SAVE OP CODE IN R7
003437  033560  010003  042420          BIS\T   #ST.OFL,R0      %JNZRO  U.C2EP  ; IF NONE THEN OFFLINE/CHANGE TO END PKT[rae03]
```



```
003440  010545  007022  103222          ADD     #SDI.PQ,UBAR\N,BAR %JMP U.LNKP  ; GO LINK UP PKT/RETURN                   ;[E121]

                                ; ***   ENTRY POINT #2 - ACTIVE PACKET PROCESSING ***

                                U.ONLB:
                                                                                        ; in case we can't find the SDI block   [rae11]
                                                                                        ; this time, ie the drive has stopped   [rae11]
                                                                                        ; clocking, and must exit, clear up     [rae11]
                                                                                        ; the SDI.ST word, but save it and      [rae11]
                                                                                        ; restore it if infact we still do have [rae11]
                                                                                        ; the SDI block                         [rae11]
003441  013440  000000  125473          NOP                             %CALL P.LOCK    ; lock out the Dproc while              [rae11]
003442  010545  007000  125572          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDQQ ; get current SDI status                [rae11]
003443  013740  007264  125613          MOV     #MSCPLN,BAR             %CALL S.STQQ    ; and save it in temp storage           [rae11]
003444  103740  000034  010000          MOV     #<DRAVL!DRVOL!DRDUP>,Q                  ; clear all but drive status            [rae11]
003445  010545  007000  000000          ADD     #SDI.ST,UBAR\N,BAR                      ;                                       [rae11]
003446  004600  000003  135613          AND     (BUF),Q         %CALL   S.STQQ  ; and set back into SDI status          [rae11]
003447  033442  000010  123711          MOV     R10,R2          %CALL   U.ULKA  ; temp unlink from SDI.PQ               [rae05]
003450  033450  000002  010000          MOV     R2,R10                          ; restore MSCP packet adrs              [rae05]
003451  033447  000000  000000          MOV     R0,R7                           ; save R0 to R7 earlier                 [rae05]
003452  013440  000000  122551          NOP                             %CALL   U.GSDI  ; refind SDI cont1 block                [rae05]
003453  013440  010000  113456          NOP                             %JZRO   10$     ; if there is none now, error           [rae11]
003454  013440  000000  135474          NOP                             %CALL   UNLOCK  ;   unlock the dproc and then          [rae11]
003455  033540  000003  102420          BIS     #ST.OFL,R0      %JMP    U.C2EP  ;   go create the end packet            [rae11]
                                10$:                                                    ;                                       [rae11]
003456  013440  000000  133230          NOP                             %CALL   U.LNKH  ; now relink to SDI.PQ                  [rae05]
003457  013740  007264  125575          MOV     #MSCPLN,BAR     %CALL   S.LDR2  ; now restore the SDI status            [rae11]
003460  010545  007000  125621          ADD     #SDI.ST,UBAR\N,BAR %CALL S.STR2 ; and set it into R2                    [rae11]
003461  013440  000000  135474          NOP                             %CALL   UNLOCK  ; and unlock the dproc now              [rae11]
                                ;       ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR2 ; R2=SDI STATUS
003462  014542  000200  000000          BIT     #XCMP,R2                        ; IF TRANSFER COMPLETE
003463  013440  010000  003546          NOP                             %JNZRO  U.ONLF  ;                                       [rae05]
                                ;       MOV     R0,R7           %JNZRO  U.ONLF  ;   THEN CONTINUE                       [rae05]
003464  033746  000040  122610          MOV     #SUSP,R6        %CALL   U.GUCP  ; GO GET UNIT CHAR PTR
003465  033740  010003  113641          MOV     #ST.OFL,R0      %JZRO   U.DONE  ; IF NOT THERE THEN MUST HAVE GONE OFFLINE
003466  033741  000020  124321          MOV     #MD.SHD,R1      %CALL   U.TMOD  ; GO TEST IF SHADOW MODIFIER PRESENT
003467  033761  010005  052502          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDE  ; IF SET THEN INVALID COMMAND
003470  016547  000012  000000          XOR     #OP.SUC,R7\N                    ; IF SET UNIT CHAR CMD
003471  114640  010010  113504          BIT     #DRV.AV,Q       %JZRO   U.ONLS  ;   THEN SET UNIT FLAGS
003472                                  ASSUME  ST.SUC,EQ,0                     ; MAKE SURE ST.SUC IS ZERO
003472  133760  010001  153500          MOV\T   #SC.AOL,R0      %JZRO   U.ONLZ  ; IF UNIT ONLINE THEN FINISH UP
003473  010545  007044  135575          ADD     #SDI.S1,UBAR\N,BAR %CALL S.LDR2 ; R2=DRIVE STATUS
003474                                  ASSUME  DRV.RU,EQ,BIT00                 ; MAKE SURE DRV.RU IS LSB
003474  114542  040010  113500          BIT     #DRV.DD,R2      %JNLSB  U.ONLZ  ; IF NOT SPINABLE THEN SET STATUS/QUICK EXIT
003475  013440  010000  103505          NOP                             %JZRO   U.ONLC  ; IF NOT DISABLED THEN CONTINUE
003476  033741  000001  124321          MOV     #MD.RIP,R1      %CALL   U.TMOD  ; IF PKT MODIFIERS ALLOW SELF-DESTRUCT
003477  033740  010003  053505          MOV\F   #ST.OFL,R0      %JNZRO  U.ONLC  ;   THEN CONTINUE/ELSE R0=OFFLINE STATUS

                                ; ***   SET STATUS IN R0 IN SDI.SW, THEN GIVE CHARACTERISTICS AT U.ONLL ***

003500  010545  007003  125574  U.ONLZ: ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDR1 ; BAR=PTR TO MSCP STATUS[mjto6] make sure nothing else is
003501  014541  000017  010000          BIT     #17,R1                          ; [mjto6]DO A BIT TEST OF THE ERROR
003502  033460  010001  043631          MOV\T   R1,R0           %JNZRO  U.ONLL  ; [mjto6]IF NOT ZERO, RELOAD R0 AND BRANCH
003503  013440  003000  103631          MOV     R0,BUF          %JMP    U.ONLL  ;   ELSE UNSPINABLE JUST GIVE INFO
```

```
                                ; ***   SET UNIT CHARACTERISTICS COMMAND - CHECK IF UNIT ONLINE ***

003504  033760  010004  053500  U.ONLS: MOV\T    #ST.AVL,R0      %JNZRO  U.ONLZ  ; IF AVAILABLE THEN JUST GIVE INFO

                                ; ***   RESET SUBUNIT FLAGS ***

003505  033742  000003  000000  U.ONLC: MOV     #UF.MSK,R2                      ; R2=UNIT FLAG MASK
003506  010550  007011  125573          ADD     #P.UNFL,R10\N,BAR %CALL S.LDR0  ; R0=NEW UNIT FLAGS
003507  033741  000004  124321          MOV     #MD.SWP,R1      %CALL   U.TMOD  ; IF WRITE PROTECT MODIFIER
003510  133562  010020  043511          BIS\T   #UF.WPS,R2      %TNZRO          ;   THEN SET SOFTWARE WRITE PROTECT
003511  034140  000002  010000          AND     R2,R0                           ; ISOLATE FLAGS CONTROLLER LIKES
003512  010545  007060  125574          ADD     #SDI.UF,UBAR\N,BAR %CALL S.LDR1 ;[V05] R1=EXISTING SUBUNIT FLAGS
003513  035141  000002  010000          BIC     R2,R1                           ; CLEAR FLAGS HOST CAN SET
003514  016547  000012  000000          XOR     #OP.SUC,R7\N                    ; IF SET UNIT CHAR COMMAND
003515  033140  013001  102375          OR      R1,R0,BUF       %JZRO   U.CLRS  ;   THEN CLEAR SUSP/EXIT-OR INTO UNIT FLAGS & RESET

                                ; ***   CALCULATE 1ST COPY XBN NUMBER ***

003516  103740  000240  000000          MOV     #XBNCOD,Q                       ; Q=XBN HEADER CODE
003517  034441  000001  113527          CLR     R1              %JMP    U.ONLE  ; ZAP LO STARTING XBN/CONTINUE

                                ; ***   UPDATE XBN TO NEXT COPY STARTING XBN ***

003520  033441  010003  113554  U.ONLD: MOV     R3,R1           %JZRO   U.ONLX  ; IF ALL COPIES TRIED THEN ERROR
003521  013440  000000  122610          nop                     %call   u.gucp  ; Restore the unit characteristics' pointer [ch01]
003522  010545  007073  125573          ADD     #SDI.FC,UBAR\N,BAR %CALL S.LDR0 ;[V05] R0=FCT COPY SIZE
003523  010550  007020  125574          ADD     #P.LBNO,R10\N,BAR %CALL S.LDR1  ; R1=LO PREVIOUS START XBN
003524  010550  007021  135572          ADD     #P.LBN1,R10\N,BAR %CALL S.LDQQ  ; Q=HI PREVIOUS START XBN
003525  030141  000000  000000          ADD     R0,R1                           ; ADD IN FCT SIZE
003526  100260  020000  143527          INC\T   Q               %TCRY           ; IF CARRY THEN UPDATE HI XBN

                                ; ***   STORE STARTING XBN IN PACKET ***

003527  010550  007020  135620  U.ONLE: ADD     #P.LBNO,R10\N,BAR %CALL S.STR1  ; SAVE LO XBN
003530  010550  007021  125613          ADD     #P.LBN1,R10\N,BAR %CALL S.STQQ  ; SAVE HI XBN

                                ; ***   FORGE TRANSFER COUNT SO U.ALOC WILL ONLY ALLOCATE 1 BUFFER ***

003531  010545  007015  010000          ADD     #SDI.XL,UBAR\N,BAR              ; BAR=PTR TO LO TRANSFER COUNT
003532  113740  003002  132473          MOV     #SECSZ*2,BUF    %CALL   U.CSPT  ; SET IN 1 SECTOR'S WORTH/GO CALC SECS/TRK
003533  033440  000002  000000          MOV     R2,R0                           ; R0=# SECTOR'S PER TRACK
003534  010545  007016  135630          ADD     #SDI.XH,UBAR\N,BAR %CALL S.CLRB ; ZAP HI ORDER BYTES REMAINING

                                ; ***   GET LBN CYLINDERS ***

003535  010545  007063  135601          ADD     #SDI.H1,UBAR\N,BAR %CALL S.LDR7 ;[V05] R7=LBN CYLS LO
003536  010545  007064  135600          ADD     #SDI.H2,UBAR\N,BAR %CALL S.LDR6 ;[V05] R6=LBN CYLS HI + HI CYL NUMBER
003537                          ASSUME  <CNVTV1&17>,LE,<17-<CNVTV4-CNVTV1>>; USE AUTO-INC TO PAINT CNVTVX
003537  013740  007040  135625          MOV     #CNVTV1,BAR     %CALL   S.STR7  ; SAVE LO LBN CYLS
003540  013440  003606  135620          MOV     R6,BUF  @RUPF   %CALL   S.STR1  ; SAVE HI LBN CYLS/SAVE LO XBN
003541  135642  000360  132764          BIC     #XBNSTR,Q,R2    %CALL   U.IOPY  ; ZAP XBN CODE FROM HI XBN

                                ; ***   SET SEEK REQUESTED BIT IN SDI STATUS ***

KDBUP           KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```



```
003542                          ASSUME  ST.SUC,EQ,0                    ; MAKE SURE SUCCESS IS ZERO
003542  010545  007003  125630          ADD     #SDI.SW,UBAR\N,BAR %CALL S.CLRB ; MAKE STATUS=SUCCESS OPTIMISTICALLY
003543  010550  007024  125601          ADD     #S.CYLL,R10\N,BAR %CALL S.LDR7  ; R7=LO XBN CYL NUMBER
003544  010550  007025  125606          ADD     #S.CYLH,R10\N,BAR %CALL S.LLB6  ; R6=HI XBN CYL NUMBER
003545  013440  000000  103034          NOP                     %JMP    U.IOPZ  ;[U52EC2] BYE-BYE SUBUBITS...

                                ; ***   SET UNIT CHAR DONE, ERROR OR PSEUDO AVAILABLE ENTRY POINT ***

003546  010545  007003  135576  U.ONLF: ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDR3 ; R3=ERROR STATUS
003547  016547  000012  000000          XOR     #OP.SUC,R7\N                    ; IF SET UNIT CHAR CMD
003550  014542  010004  113631          BIT     #DERR,R2        %JZRO   U.ONLL  ;   THEN CONTINUE/IF NOT ERROR
003551  016543  010110  103631          XOR     #<ST.DAT!SC.HDR>,R3\N %JZRO U.ONLL; [U52EC1]THEN DONE/IF HDR NOT FOUND
003552  111543  010137  153564          SUB\F   #<ST.MSK+SC.INV>,R3\N %JZRO U.ONLY; THEN CONTINUE/ELSE GET SUB-CODE
003553  033441  020003  003640          MOV     R3,R1           %JNCRY  U.DONX  ; IF LT/EQ SC.INV THEN FATAL SDI LEV2 & DONE

                                ; ***   MARK UNIT AVAILABLE SO D.PROC WILL DISCONNECT ***

003554  010545  007003  135620  U.ONLX: ADD     #SDI.SW,UBAR\N,BAR %CALL S.STR1 ; SAVE ERROR CODE
003555  010550  007001  125574          ADD     #P.VCID,R10\N,BAR %CALL S.LDR1  ; [U52EC3]R1=VIRT CIRCUIT ID & FLAGS
003556  014541  000002  000000          BIT     #PONER,R1                       ; [U52EC3]IF ERROR ON AVAILABLE SEQ. AFTER ONLINE FAILURE
003557  033541  013002  053640          BIS\F   #PONER,R1,BUF   %JNZRO  U.DONX  ; [U52EC3]  THEN FATAL ERR AND DONE/ELSE SET #PONER
003560  010550  007007  000000          ADD     #P.MOD,R10\N,BAR                ; BAR=PTR TO PKT MODIFIERS
003561  013740  003001  135476          MOV     #MD.SPD,BUF     %CALL   U.GMST  ; R0=MUTEXED SDI STATUS
003562  035540  003044  105474          BIC     #<SUSP+DERR>,R0,BUF %JMP UNLOCK ; CLR SO ONLINE CAN WORK AGAIN
                                ;[mjt06]BIC     #<SUSP+XCMP+DERR>,R0,BUF %JMP UNLOCK ; CLR SO ONLINE CAN WORK AGAIN

                                ; ***   FCT READ COMPLETE ENTRY POINT ***
                                ; ***   CHECK FOR ECC ERROR ***

003563  033746  000100  122375  U.ONLR: MOV     #BFSV,R6        %CALL   U.CLRS  ; CLEAR BUFFER SERVICE
003564  010545  007006  135601  U.ONLY: ADD     #SDI.UB,UBAR\N,BAR %CALL S.LDR7 ; R7=BUFFER POINTER
003565  010547  007001  135573          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR0  ; [16K]R0=BUFFER STATUS WORD
003566  033741  000002  134320          MOV     #MD.IMF,R1      %CALL   U.TMDA  ; IF IGNORE MEDIA FORMAT ERROR MODIFIER SET
003567  034460  010000  053500          CLR\T   R0              %JNZRO  U.ONLZ  ;   THEN HE DOESN'T CARE/MAKE STATUS EQ 0
003570  114540  000020  000000          BIT     #BECC,R0                        ; IF ECC ERROR
003571  013460  010000  076450          NOP\T                   %CNZRO  ECCC    ;   THEN CALL ECC ROUTINE
003572  033743  010345  143607          MOV\F   #<ST.MFE+SC.ECC>,R3 %JZRO U.ONLI; IF CORRECTED THEN CONTINUE

                                ; ***   UNCORRECTABLE ECC OR EDC ERROR - INVOKE RECOVERY ***

003573  010545  007034  135574  U.ONLG: ADD     #SDI.RC,UBAR\N,BAR %CALL S.LDR1 ; R1=FCT COPIES
003574  010550  007010  135573          ADD     #P.RSO6,R10\N,BAR %CALL S.LDR0  ; R0=COPY LEVEL
003575                          ASSUME  ONLREC,EQ,BIT15                 ; MAKE SURE ONLINE RECOVERY EQ MSB
003575  134541  030017  103600          AND     #LONIB*256.,R1  %JMSB   U.ONLH  ; ISOLATE FCT COPIES
003576  003740  000010  135501          MOV     #8.,Q           %CALL   S.RRR1  ; SWAP BYTES OF COPIES
003577  133541  003200  010000          BIS     #ONLREC,R1,BUF                  ; ACTIVATE COPY COUNT

                                ; ***   INVOKE LEVEL 1 ERROR RECOVERY ***

003600  013440  000000  134731  U.ONLH: NOP                     %CALL   U.RREC  ; GO DO LEVEL 1 ERROR RECOVERY
003601  033746  010204  037777          MOV     #<XCMP!DERR>,R6 %RNZRO          ; IF STARTED THEN RET
003602  133546  000100  122375          BIS     #ERRIP,R6       %CALL   U.CLRS  ; ZAP DRIVE ERR,XCMP & LEV 1 ERR
003603  010545  007017  125630          ADD     #SDI.E1,UBAR\N,BAR %CALL S.CLRB ; CLEAR LEVEL 1 ERROR RECOVERY

KDBUP           KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```

```
003604  010550  007010  125614          ADD     #P.RS06,R10\N,BAR %CALL S.DECB  ; GO DECR R0 AND RESET
003605  013440  000000  125505          NOP                     %CALL   S.RELC  ; [16K]RELEASE THE BUFR CNTRL BLKS
003606  034540  000017  113520          AND     #LONIB,R0       %JMP    U.ONLD  ; ISOLATE COUNT

                                ; ***   CALCULATE EDC AND VERIFY ***

003607  016543  000110  010000  U.ONLI: XOR     #<ST.DAT!SC.HDR>,R3\N           ; IF HEADER COMPARE ERROR
003610  003740  010336  103573          MOV     #MD512&LOBYT,Q  %JZR0   U.ONLG  ;   THEN DO ERROR RECOVERY
003611  010547  007012  135630          ADD     #BUF.BC,R7\N,BAR %CALL  S.CLRB  ; ZAP BYTE COUNT

003612  010547  007013  125630          ADD     #BUF.U1,R7\N,BAR %CALL  S.CLRB  ; CLEAR LOWER ADR
003613  010547  007014  135630          ADD     #BUF.U2,R7\N,BAR %CALL  S.CLRB  ; CLEAR UPPER ADR AND COUNT
003614  010547  007011  135574          ADD     #BUF.GP,R7\N,BAR %CALL  S.LDR1  ; [mjt06] get SI size in group
003615  034541  003377  000000          AND     #LOBYT,R1,BUF                   ; [mjt06] and strip off
003616  010547  007002  135575          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR2  ; [16K]R2=BUFFER PTR
003617  033743  000105  125175          MOV     #69.,R3         %CALL   U.CKED  ; [16K]R3=EDC START #/GO CALC EDC
003620  016503  000003  000000          XOR     (BUF),R3\N                      ; IF EDC'S NEQ
003621  033763  010112  043573          MOV\T   #<ST.CNT+SC.EDC>,R3 %JNZR0 U.ONLG;    THEN DO RECOVERY

                                ; ***   EVERYTHING COOL - CHECK PACK FORMAT ***

003622  010547  007002  135576          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR3  ; [16K]R3=FCT DATA BUFFER PTR
003623  133640  000255  125505          OR      #MD512&HIBYT,Q,R0 %CALL S.RELC  ; [16K]TEST FOR HI BYTE OF MODE CODE
003624  010543  007000  135610          ADD     #FCT.MD,R3\N,BAR %CALL  S.XOR0  ; [16K]GO XOR R0 WITH (BUF)
003625  033761  010245  043554          MOV\T   #<ST.MFE+SC.N12>,R1 %JNZR0 U.ONLX ; IF NOT IN 512-BYTE FORMAT THEN ERROR

                                ; ***   PUT VOLUME IDENTIFIER INTO SUBUNIT CHARACTERISTICS BLOCK ***

003626  000545  000060  000000          ADD     #<SDI.V1-1>,UBAR,Q              ; [U52EC2]Q=START OF SEQUENTIAL STORE AREA;[E121]
003627  010543  007002  133365          ADD     #FCT.V1,R3\N,BAR %CALL  STBFLG  ; [U52EC2]GET AND SAVE VOLUME ID LO
003630                                  ASSUME  SDI.V2,EQ,SDI.V1+1              ; MAKE SURE SEQUENTIAL
003630  010543  007003  123365          ADD     #FCT.V2,R3\N,BAR %CALL  STBFLG  ; [U52EC2]GET AND SAVE VOLUME ID HI

                                ; ***   SET UNIT FLAGS, MULTI-UNIT CODE, UNIT ID, ETC. ***

003631  033443  000005  133702  U.ONLL: MOV     UBAR,R3         %CALL   U.SUNI  ; SET IN UNIT,MEDIA TYPE ID'S

                                ; ***   FILL IN UNIT SIZE (IN LBN'S) ***

003632                                  ASSUME  <P.UNS0-1>,EQ,P.SHST            ; MAKE SURE Q SET UP RIGHT
003632  010545  007075  133365          ADD     #SDI.L1,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND SAVE LBN'S IN HOST AREA (LO)
003633                                  ASSUME  P.UNS1,EQ,P.UNS0+1              ; MAKE SURE SEQUENTIAL
003633  010545  007076  133365          ADD     #SDI.L2,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND SAVE LBN'S IN HOST AREA (HI)
003634                                  ASSUME  P.VSE0,EQ,P.UNS1+1              ; MAKE SURE SEQUENTIAL

                                ; ***   FILL IN VOLUME SERIAL NUMBER ***

003634  010545  007061  133365          ADD     #SDI.V1,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND SAVE VOL SERIAL NUMBER LO
003635                                  ASSUME  P.VSE1,EQ,P.VSE0+1              ; MAKE SURE SEQUENTIAL
003635  010545  007062  133365          ADD     #SDI.V2,UBAR\N,BAR %CALL STBFLG ;[V05] R0=VOL SERIAL NUMBER HI

                                ; ***   MAKE SURE LAST TWO WORDS ARE ZERO ***

003636  010550  007030  135630          ADD     #P.RCTS,R10\N,BAR %CALL S.CLRB  ; ZAP WORD 22
```



```
003637  010550  007031  125630          ADD     #P.RBNS,R10\N,BAR %CALL S.CLRB  ; ZAP WORD 23

                                ; ***   ENTRY POINT #3 - ONLINE ERROR EXIT POINT ***
                                ; ***   GENERAL EXIT POINT FOR ALL MSCP TERMINATIONS ***

003640  010545  007003  135573  U.DONX: ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDR0 ; R0 = APPROPRIATE STATUS CODE

                                ; ***   PACKET COMPLETE EXIT - R0 HAS STATUS CODE ***

003641  033446  000010  133711  U.DONE: MOV     R10,R6          %CALL   U.ULKA  ; SAVE R10 (PKT PTR) IN R6
003642  033450  000006  132377          MOV     R6,R10          %CALL   U.C2ED  ; GO CHANGE TO END PKT
003643  013440  000000  125505          NOP                     %CALL   S.RELC  ; [16K]RELEASE ALL BUFFERS(JUST IN CASE)
003644  010545  007025  135600          ADD     #SDI.OM,UBAR\N,BAR %CALL S.LDR6 ; [U52EC1]IF CMD WAS OVERLAPPED

                                ; ***   EXIT POINT TO CLEAR SDI STATUS AND DEACTIVATE A PACKET ***

003645  010545  007003  125630  U.CSTA: ADD     #SDI.SW,UBAR\N,BAR %CALL S.CLRB ; [U52EC1]ZAP STATUS
003646  013440  000000  125476          NOP                     %CALL   U.GMST  ; GET MUTEXED SDI STATUS IN R0
003647  013446  000006  000000          TST     R6                              ; [U52EC1]IF R6(SDI.OM) EQ 0
003650  134560  013034  145474          AND\T   #<DRVOL+DRAVL+DRDUP>,R0,BUF %JZR0 UNLOCK; [U52EC1]  THEN ZAP STATUS/FREE D.PROC/RETURN
003651  035540  003304  125474          BIC     #<XCMP!BFSV!DERR>,R0,BUF %CALL UNLOCK; [U52EC1]  ELSE JUST ZAP XCMP & BFSV/RETURN
003652  010545  007025  105630          ADD     #SDI.OM,UBAR\N,BAR %JMP S.CLRB  ; [U52EC1]ZAP SDI.OM/RETURN
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SCCH (MSCP SET CONTROLLER CHARACTERISTICS)
                                ;       'IMMEDIATE COMMAND'
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE MSCP 'SET CONTROLLER
                                ; CHARACTERISTICS' COMMAND FOR THE CONTROLLER.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP  PACKET
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET QUEUED TO THE HOST.
                                ;
                                ;;-

                                ; ***   VERIFY MSCP VERSION NUMBER ***

000001                          M.SCCL  :=  MD.FKC                                      ; [ch10] Modifier allowed for OP.SCC
                                U.SCCH:
003653  003740  000001  122263          MOV     #M.SCCL,Q       %CALL   TSTMOD  ; [ch10] test for reserved modifier bits    [rae02]
003654  033761  010005  042506          MOV\T   #<P.MOD-P>,R1   %JNZRO  U.CMDA  ;   it set then error                      [rae02]
003655  013440  000000  122255          NOP                     %CALL   MLN40   ; test that msg leng is large enough [rae02]
003656  034441  010001  002506          CLR     R1              %JNZRO  U.CMDA  ;   if not declare error                  [rae02]
003657  010550  007010  135573          ADD     #P.VRSN,R10\N,BAR %CALL S.LDRO  ; R0=MSCP VERSION NUMBER
003660  033761  010006  042506          MOV\T   #<P.VRSN-P>,R1  %JNZRO  U.CMDA  ; IF NEQ 0 THEN INVALID COMMAND

                                ; ***   SET HOST TIMEOUT ***

003661  010550  007012  125573          ADD     #P.HTMO,R10\N,BAR %CALL S.LDRO  ; R0=HOST TIMEOUT IN SECS
003662  013740  007200  135616          MOV     #HOSTMO,BAR     %CALL   S.STRO  ; SAVE HOST TIMEOUT
003663  013740  007174  135616          MOV     #TMR.MC,BAR     %CALL   S.STRO  ; START HOST TIMER
003664  000550  000011  010000          ADD     #<P.CTMO-1>,R10,Q               ; Q=START OF SEQUENTIAL STORE AREA

                                ; ***   SET CONTROLLER TIMEOUT ***

003665  033740  000170  123370          MOV     #CTMOUT,R0      %CALL   STROLG  ; STORE CONTROLLER TIMEOUT

                                ; ***   SET CONTROLLER HARDWARE/SOFTWARE VERSION NUMBERS ***

003666                          ASSUME  P.CSVR,EQ,P.CTMO+1                      ; MAKE SURE SEQUENTIAL
003666  033740  000023  127750          MOV     #CODVER,R0      %CALL   U.HWVR  ; R0=CONTROLLER SW VERSION NUMBER /
                                                                                ;  SET IN HW VERSION NUMBER.

                                ; ***   SET CONTROLLER ID NUMBER ***

003667  000550  000013  137751          ADD     #<P.CNTO-1>,R10,Q %CALL U.SCID  ; GO SET CONTROLLER ID

                                ; ***   SET CONTROLLER FLAGS ***

003670  010550  007011  125573          ADD     #P.CNTF,R10\N,BAR %CALL S.LDRO  ; R0=CONTROLLER FLAGS
003671  034540  003320  000000          AND     #CFLAGS,R0,BUF                  ; ISOLATE FLAGS THAT CONTROLLER SUPPORTS
003672  013740  007170  135574          MOV     #CNTFLG,BAR     %CALL   S.LDR1  ; R1=OLD FLAGS
```



```
                                ; ***   NOTE - HI BYTE OF CNTFLG IS CON.ST (VAX PURGE,ERROR INTERRUPT FLAGS,ETC) ***

003673  134541  000377  010000          AND     #HIBYT,R1                       ; CLEAR OUT OLD CONTROLLER FLAGS
003674  033141  003000  102303          OR      R0,R1,BUF       %JMP    U.IMEX  ; SET IN NEW ONES/TAKE IMMED CMD EXIT
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SUID (SET UNIT ID)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SET THE UNIT IDENTIFIER IN THE AREA
                                ; POINTED TO BY REGISTER Q.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;       Q               POINTER TO UNIT ID STOREAGE AREA
                                ;
                                ; OUTPUTS:
                                ;       UNIT ID IN AREA POINTED TO BY Q
                                ;
                                ;;-
003675  010545  007037  133365  U.SUID: ADD     #SDI.I1,UBAR\N,BAR %CALL STBFLG ; GET AND STORE UNIT ID #1
003676                                  ASSUME  P.UNT1,EQ,P.UNTO+1              ; MAKE SURE SEQUENTIAL
003676  010545  007040  133365          ADD     #SDI.I2,UBAR\N,BAR %CALL STBFLG ; GET AND STORE UNIT ID #2
003677                                  ASSUME  P.UNT2,EQ,P.UNT1+1              ; MAKE SURE SEQUENTIAL
003677  010545  007041  123365          ADD     #SDI.I3,UBAR\N,BAR %CALL STBFLG ; GET AND STORE UNIT ID #3
003700  010545  007042  135604          ADD     #SDI.TI,UBAR\N,BAR %CALL S.LLBO ; RO=DRIVE TYPE ID (LO BYTE)
003701  133540  000002  103370          BIS     #DCLASS,RO        %JMP   STROLG ; SET IN DISK CLASS/STORE/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.SUNI (SET UNIT ID, MEDIA TYPE, FLAGS, ETC.)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL SET THE UNIT FLAGS, MULTI-UNIT CODE,
                                ; THE UNIT IDENTIFIER, THE MEDIA TYPE IDENTIFIER, THE SHADOW STATUS,
                                ; AND THE SHADOW UNIT INTO THE END PACKET POINTED TO BY R10. THIS
                                ; FORMAT IS USED BY MSCP 'GET UNIT STATUS', 'ONLINE', AND 'SET UNIT
                                ; CHARACTERISTICS' COMMANDS.
                                ;
                                ; INPUTS:
                                ;       R10             POINTER TO MSCP END PACKET
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       MSCP 'END' PACKET WITH ABOVE DESCRIBED FIELDS FILLED IN
                                ;
                                ;;-
003702  000550  000007  123401  U.SUNI: ADD     #<P.MLUN-1>,R10,Q %CALL U.MLUN  ; Q=START OF SEQUENTIAL STORE AREA

                                ; ***   SET UNIT FLAGS AND UNIT ID FIELDS ***

003703                                  ASSUME  P.UNFL,EQ,P.MLUN+1              ; MAKE SURE SEQUENTIAL
003703  010545  007060  123365          ADD     #SDI.UF,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND STORE UNIT FLAGS

                                ; ***   FILL IN UNIT IDENTIFIER ***

003704  000550  000013  133675          ADD     #<P.UNTO-1>,R10,Q %CALL U.SUID  ; Q=STORE AREA/GO SET UNIT ID

                                ; ***   FILL IN MEDIA TYPE IDENTIFIER ***

003705                                  ASSUME  <P.MED1-1>,EQ,P.UNT3            ; MAKE SURE Q SET UP RIGHT
003705  010545  007071  123365          ADD     #SDI.M1,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND STORE 1ST WORD OF MEDIA TYPE
003706                                  ASSUME  P.MED2,EQ,P.MED1+1              ; MAKE SURE SEQUENTIAL
003706  010545  007072  123365          ADD     #SDI.M2,UBAR\N,BAR %CALL STBFLG ;[V05] GET AND STORE 2ND WORD OF MEDIA TYPE

                                ; ***   FILL IN SHADOW UNIT AND SHADOW UNIT STATUS ***

003707                                  ASSUME  P.SHUN,EQ,P.MED2+1              ; MAKE SURE SEQUENTIAL
003707  034440  000000  133370          CLR     RO                %CALL   STROLG ; zero Shadow unit (new MSCP)    ;[E121]
003710                                  ASSUME  P.SHST,EQ,P.SHUN+1              ; MAKE SURE SEQUENTIAL
003710  034440  000000  103370          CLR     RO                %JMP    STROLG ; ZAP SHADOW STATUS/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.ULNK  (UNLINK PACKET FROM QUEUE HEAD)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL UNLINK A PACKET FROM A QUEUE HEAD PASSED
                                ; IN R7 AND BAR.
                                ;
                                ; INPUTS:
                                ;       R7,BAR          POINTER TO QUEUE HEAD
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK (U.ULKA ENTRY)
                                ;
                                ; OUTPUTS:
                                ;       CURRENT PACKET RELEASED
                                ;       NEXT PACKET REMOVED FROM QUEUE
                                ;       R10             NONZERO = POINTER TO NEXT PACKET
                                ;                       ZERO    = NO MORE PACKETS IN QUEUE
                                ;
                                ;;

003711  033447  000005  000000  U.ULKA: MOV     UBAR,R7                         ; R7=SDI CONTROL BLK PTR
003712  030547  007022  000000          ADD     #SDI.PQ,R7,BAR                  ; BAR,R7=QUEUE HEAD POINTER
003713  033710  000003  010000  U.ULNK: MOV     (BUF),R10                       ; R10=PTR TO QUEUE HEAD
003714  013440  007010  135602          MOV     R10,BAR         %CALL   S.LD10  ; R10=PKT STATUS & NEXT LINK PTR
003715  013740  003000  000000          MOV     #0,BUF                          ; FREE THIS PACKET
003716  115550  000300  010000          BIC     #Q.STAT,R10\N                   ; ISOLATE NEXT LINK FIELD
003717  034470  010010  143720          CLR\T   R10             %TZRO           ; IF EQ 0 THEN LAST PKT
003720  013440  007007  105626          MOV     R7,BAR          %JMP    S.ST10  ; RESET QUEUE HEAD
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.STUD (SERVER TASK FOR UTILITIES AND DIAGNOSTICS, AKA DUP)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE INITIAL VALIDATION OF THE DUP COMMAND
                                ; AND VECTORS INTO THE VECTOR TABLE TO PROCESS EACH LEGAL COMMAND.
                                ;
                                ; INPUTS:
                                ;       CC=ZERO IF VIRTUAL CIRCUIT IS DUP (DIAGNOSTIC/UTILITIES PROTOCOL) CIRCUIT
                                ;       R0=OPCODE OF DUP PACKET
                                ;
                                ; OUTPUTS:
                                ;       DUP COMMAND IS PERFORMED, RETURN TO IDLE LOOP.
                                ;       DMODE AND DMBEG FLAGS IN RLL AND DM DATA BASE MODIFIED.
                                ;
                                ;;-

003721  013440  010000  052176  U.STUD: TST\F    R0                      %JNZRO  U.VCER  ; BR IF ILLEGAL VIRTUAL CIRCUIT
003722  011540  010006  112231          SUBC    #MAXSTC,R0\N            %JZRO   U.CMER  ; IF OP CODE EQ 0 THEN ERROR/CHK HI RANGE
003723  030540  020330  112231          ADD     #STDTBL-1&LOBYT,R0 %JCRY U.CMER         ; IF SO, ERROR; ELSE DISPATCH
003724  133540  002377  000000          BIS     #HIBYT,R0,PAR                   ; ON DUP OPCODE
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GSST (GET DUST STATUS COMMAND)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE GET DUST STATUS COMMAND
                                ;
                                ; INPUTS:
                                ;       Q               PACKET OFFSET OF P.BCN1-1
                                ;
                                ; OUTPUTS:
                                ;       DUST STATUS
                                ;
                                ;;-

                                U.GSST::ADD     #<P.BCNO-1>,R10,Q %JMP  U.GSST  ; IN VECTOR TABLE
003725  033740  000102  010000          MOV     #'B,R0                          ;
003726  133540  000104  123370          BIS     ##'D,R0         %CALL   STROLG  ; SET DEFAULT FILE EXT TO 'BDA'
003727  033740  000101  010000          MOV     #'A,R0                          ;
003730  114544  000040  010000          BIT     #DMBEG,RLL                      ; SEE IF A DM PROGRAM IS EXECUTING
003731  133560  010010  053732          BIS\T   #BTROTH,R0      %TNZRO          ; SET 'BETROTHED' STATE IF SO
003732  133540  000001  133370          BIS     #STDALN,R0      %CALL   STROLG  ; ALSO SET 'STANDALONE' BIT
003733  013440  000000  135534          NOP                     %CALL   S.FUNB  ; [16K]UBAR=DM.BEG ADDRESS
003734  010545  007006  123365          ADD     #DSTSL,UBAR\N,BAR %CALL STBFLG  ; [16K]MOVE PROGRESS INDICATOR
003735  010545  007007  133365          ADD     #DSTSH,UBAR\N,BAR %CALL STBFLG  ; [16K]FROM DM.BEG+DSTSL &DM.BEG+DSTSH
003736  033740  000000  112420          MOV     #ST.SUC,R0      %JMP    U.C2EP  ; AND RETURN SUCCESSFUL
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.EXSP (EXECUTE SUPPLIED PROGRAM)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE EXECUTE SUPPLIED PROGRAM COMMAND.
                                ;
                                ; INPUTS:
                                ;       BIT 'DMBEG' TESTED IN RLL
                                ;
                                ; OUTPUTS:
                                ;
                                ;;-

                                U.EXSP::BIT     #DMBEG,RLL              %JMP    U.EXSP  ; EXECUTED IN VECTOR TABLE
003737  133544  010120  047733          BIS\F   #<DMODE!DXFC>,RLL %JNZRO U.STER ; [16K]ERROR IF PROGRAM ALREADY EXECUTING ;[E122]
                                                                                ;  else set DMODE and DXFC then ;[E122]
003740  114544  010020  003740  1$:     BIT     #DXFC,RLL               %JNZRO  1$      ;  wait for DPROC to clr DXFC   ;[E122]
                                                                                ;  indicating that it is now    ;[E122]
                                                                                ;  really in DMODE              ;[E122]
003741  004240  000000  133373          CLR     Q                       %CALL   U.MIOA  ; [16K]CLR Q/SET UP XFER PARMS
003742  130441  000001  000000          INC     R1                              ; PREPARE FOR ADD TO (R1,IUAR)
003743  030546  000040  125540          ADD     #<OFFSET*2>,IUAR %CALL  S.AXAB  ;[BDA] BUMP UNIBUS ADDR PAST HEADER
003744  034441  000001  000000          CLR     R1                              ;[c119-rae5] clear EDC incase unb.rd is not called
003745  031540  000020  000000          SUBC    #OFFSET,R0                      ; CHECK FOR ILLEGAL BC AND ADJUST
003746  130440  030000  035635          INC     R0              %CNMSB  UNB.RD  ; BYTE COUNT FOR HEADER (NO XFER  IF NULL)
003747  033760  010011  052420          MOV\T   #ST.HST,R0      %JNZRO  U.C2EP  ; IF ILLEGAL BC OR XFER FAILURE THEN ERROR
003750  013441  000001  000000          TST     R1                              ; [qda] test edc
003751  033760  010011  052420          MOV\T   #ST.HST,R0      %JNZRO  U.C2EP  ; [qda] error if not zero
003752  010550  007014  125573          ADD     #P.BUF2,R10\N,BAR %CALL S.LDRO  ; GET ADDR OF CONSOLE TTY FOR DMDT
003753  013440  000000  135534          NOP                     %CALL   S.FUNB  ; [16K]UBAR=ADDRESS OF DM.BEG
003754  010545  007005  135616          ADD     #ODTCA,UBAR\N,BAR %CALL S.STRO  ; [16K]AND STORE IT AWAY
003755  000545  000005  000000          ADD     #<DSTSL-1>,UBAR,Q               ; [16K]Q=DUST STATUS LOC-1
003756  034440  000000  133370          CLR     R0                      %CALL   STROLG  ; [16K]INIT LO DUST STATUS
003757  034440  000000  133370          CLR     R0                      %CALL   STROLG  ; [16K]INIT HI DUST STATUS
003760  010550  007020  123365          ADD     #P.LBNO,R10\N,BAR %CALL STBFLG  ; [16K]GET & STORE LO OVERLAY ADDR
003761  010550  007021  133365          ADD     #P.LBN1,R10\N,BAR %CALL STBFLG  ; [16K]GET & STORE HI OVERLAY ADDR
003762  133544  000040  112303          BIS     #DMBEG,RLL              %JMP    U.IMEX  ; DOWNLINE LOAD DONE - START DM & EXIT
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       U.EXLP (EXECUTE LOCAL PROGRAM - NOT LEGAL ON CONTROLLER)
                                ;       U.DATA (SEND DATA AND RECEIVE DATA DUP COMMANDS)
                                ;       U.SABT (ABORT PROGRAM DUP COMMAND)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES PERFORM THE ABOVE MENTIONED DUP COMMANDS.
                                ;
                                ; INPUTS:
                                ;       BAR = XXQUE (FOR U.DATA)
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;;-

                                U.EXLP:;MOV     #ST.CMD,R0      %JMP    U.C2EP  ; EXECUTED IN VECTOR TABLE

                                U.DATA:;MOV     #XXQUE,BAR      %JMP    U.DATA  ; EXECUTED IN VECTOR TBL - XX=MR OR MW
003763  114544  000040  010000          BIT     #DMBEG,RLL                      ; VERIFY THAT A PROGRAM IS EXECUTING
003764  033760  010001  152420          MOV\T   #ST.CMD,R0      %JZR0   U.C2EP  ; IF NOT, COMMAND ERROR
003765  013440  000000  103222          NOP                     %JMP    U.LNKP  ; LINK PACKET ON QUEUE AND EXIT

                                U.SABT:;BIC     #<DXFC!DMBEG>,RLL %JMP  U.IMEX  ; [16K]EXECUTED IN VECTOR TABLE/CMD ALWAYS SUCCEEDS
                                        .PAGE
```



```
                                        .SBTTL  U.PROC BUFFER ALLOCATE, READ, AND WRITE ROUTINES
                                ;;+     09-AUG-83       UDA50-A Version 5 MICROCODE
                                ; ROUTINE NAME:
                                ;       U.CSDI (CHECK SDI STATUS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE ANALYZES THE SDI STATUS OF THE INTERCONNECT
                                ; ASSOCIATED WITH THE SDI CONTROL BLOCK POINTED TO BY UBAR.THE
                                ; FOLLOWING CHECKS/ACTIONS ARE PERFORMED:
                                ;       1. IF THE 'SLAT' FLAG IS SET THEN ENTER THE ATTENTION/LOG ROUTINE.
                                ;       2. IF THE 'RVCT' FLAG IS SET THEN ENTER THE REVECTORING DISPATCHER.
                                ;       3. IF THE 'XCMP' FLAG IS SET THEN ENTER THE TRANSFER COMPLETE ROUTINE.
                                ;       4. IF THE 'BFSV' FLAG IS SET THEN ENTER THE BUFFER SERVICE ROUTINE.
                                ;       5. IF THE 'BFRQ' FLAG IS NOT SET THEN RETURN.
                                ;            ELSE FALL INTO THE BUFFER ALLOCATION ROUTINE.
                                ;
                                ; INPUTS:
                                ;       BIT #PKIP,R1    SENSE OF PKIP BIT IN SDI STATUS
                                ;       R1              SDI CONTROL BLOCK STATUS
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK TO STATUS CHECK
                                ;
                                ; OUTPUTS:
                                ;
                                ;;-

003766  114541  010040  102101  U.CSDI: BIT     #SLAT,R1        %JZR0   U.SEK0  ; IF PKIP EQ 0 THEN GO DO SEEK ORDERING
                                                                                ; IF LOG OR ATTENTION TO SEND
003767  114541  010200  013233          BIT     #RVCT,R1        %JNZR0  U.LOGS  ;   THEN GO PROCESS/IF REVECTORING REQUIRED
003770  014541  010200  004137          BIT     #XCMP,R1        %JNZR0  U.RVCT  ;   THEN GO DO IT/IF TRANSFER COMPLETE
003771  014541  010100  005230          BIT     #BFSV,R1        %JNZR0  U.XCMP  ;   THEN GO PROCESS/IF BUFFR SERVICE
003772  014541  010020  014147          BIT     #BFRQ,R1        %JNZR0  U.BFSV  ; [16K]THEN GO PROCESS/IF BUFFER REQUEST NOT SET
003773  000545  017006  177777          ADD\F   #SDI.UB,UBAR,Q,BAR %RZR0        ; [U52EC1]  THEN RETURN/ELSE BAR=PTR TO U.PROC BCB
003774                                  ASSUME  U.ALOC,EQ,.                     ; MAKE SURE U.ALOC FOLLOWS
                                        .PAGE
```

```
                                        .SBTTL  BUFFER CONTROL BLOCK ALLOCATION ROUTINE
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.ALOC (ALLOCATE BUFFER CONTROL BLOCKS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE BUFFER CONTROL BLOCK ALLOCATION. BUFFER
                                ; CONTROL BLOCKS  WILL BE ALLOCATED FROM THE BUFFER CONTROL BLOCK LIST,
                                ; BUFFER CONTROL BLOCKS ARE FREE IF THEIR LINK POINTER (FIRST WORD) IS ZERO
                                ; OTHERWISE THEY ARE IN USE. A RING WILL BE FORMED USING UP TO 33 OF THE
                                ; BUFFER CONTROL BLOCKS.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;       Q               POINTER TO SDI.UB
                                ;
                                ; OUTPUTS:
                                ;       UPDATED SDI CONTROL BLOCK
                                ;       BUFFER CONTROL BLOCK RING
                                ;
                                ;;-

                                ; *-------------------------------------------------------------------------*
                                ; *         A     L         L          00000    CCCCCC     A     TTTTTTT  EEEEEEE
                                ; *        A A    L         L         0     0  C          A A       T     E
                                ; *       A   A   L         L         0     0  C         A   A      T     E
                                ; *       AAAAAAA L         L         0     0  C         AAAAAAA    T     EEEEEE
                                ; *       A     A L         L         0     0  C         A     A    T     E
                                ; *       A     A L         L         0     0  C         A     A    T     E
                                ; *       A     A LLLLLLL LLLLLLL    00000    CCCCCC  A     A    T     EEEEEEE
                                ; *-------------------------------------------------------------------------*

003774  033703  000003  000000  U.ALOC: MOV     (BUF),R3                                ; [U52EC1][16K]R3=PTR TO U.PROC BUFR CNTRL BLK
003775  033766  010060  042375          MOV\T   #<SUSP!BFRQ>,R6 %JNZRO  U.CLRS  ; [U52EC1][16K]IF ALREADY ALLOCATED THEN CLR BFRQ/RETURN
003776                                  ASSUME  BUF.NL,EQ,0                     ; MAKE SURE BUF.NL IS ZERO
003776  013740  007256  000000          MOV     #TEMP,BAR                       ; set up to init current buf ctl blk ptr;[E121]

                                ; ***   CHECK FOR ZERO BYTE COUNT (SEEK,NO TRANSFER, AND SUCCESS) ***

003777  013740  003000  135224          MOV     #0,BUF          %CALL   U.CKTC  ; CLR BCB PTR, CHECK XFER COUNT         ;[E121]
004000  114540  010377  014005          BIT     #HIBYT,R0       %JNZRO  10$     ; [U52EC1]IF 0 THEN DO COMPLETION PROCESSING;[E121]

                                ; ***   SEEK ONLY, MUST WAIT FOR SEEK TO COMPLETE ***

004001  010545  007000  135573          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR0 ; [U52EC1]R0=SDI STATUS
004002  014540  000002  010000          BIT     #SEEK,R0                        ; [U52EC1]IF SEEK STILL GOING
004003  013440  010000  027777          NOP                     %RNZRO          ; [U52EC1]  THEN RETURN
004004  034440  000000  114472          CLR     R0              %JMP    U.XCMA  ; [U52EC1]  ELSE DONE/GO FINISH COMMAND

004005                                  ASSUME  <SECSZ*2),EQ,512.                                               ;[E121]
                                                                                ; IF XFER CNT < 2**24                   ;[E121]
004005  133741  010200  164127          MOV\F   #BIT15,R1       %CZRO   DIV512  ; THEN R1=QUO,R0=REM/ELSE SET BIG QUO   ;[E121]
004006  130461  010001  054007          INC\T   R1              %TNZRO          ; IF REMAINDER > 0 THEN INCR R1         ;[E121]
                                                                                                                        ;[E121]
```



```
                                ; ***   CALCULATE NUMBER OF SECTORS BEFORE SEEK REQUIRED                         ;[E121]
                                ; ***   SECTORS UNTIL SEEK = MIN(SECTORS IN REQ,                                 ;[E121]
                                ; ***   [(GROUPS/CYL-CUR.GROUP)*TRKS/GROUP-CUR.TRK]*SECS/TRK] - CUR.SEC)        ;[E121]
                                                                                                                ;[E121]
004007  010545  007022  135602          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; R10=MSCP PKT PTR              ;[E121]
004010  010550  007030  125575          ADD     #S.SECS,R10\N,BAR %CALL S.LDR2  ; R2 = starting sector          ;[E121]
004011                                  ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE     ;[E121]
004011  010545  007074  125606          ADD     #SDI.12,UBAR\N,BAR %CALL S.LLB6 ; R6 = LBN's/track              ;[E121]
004012  131146  000002  000000          SUB     R2,R6                           ; R6 = LBN's left in track      ;[E121]
004013  112141  000006  010000          CMP     R6,R1                           ; compare against secs in req   ;[E121]
004014  033467  020001  144027          MOV\T   R1,R7           %JCRY   20$     ; if >=, req fits on track -    ;[E121]
                                                                                ; avoid lengthier computation   ;[E121]
004015  010545  007065  125606          ADD     #SDI.GC,UBAR\N,BAR %CALL S.LLB6 ;[V05] R6=GROUPS/CYL
004016  010550  007026  135573          ADD     #S.GRUP,R10\N,BAR %CALL S.LDR0  ; R0=CUR.GROUP
004017  131146  000000  010000          SUB     R0,R6                           ; R6=GROUP/CYL-CUR.GROUP
004020  010545  007066  125560          ADD     #SDI.TG,UBAR\N,BAR %CALL MULBYT ;[V05] R7=R6*TRKS/GROUP
004021  010550  007027  135606          ADD     #S.TRAK,R10\N,BAR %CALL S.LLB6  ; R6=CUR.TRACK (hi byte has SDI op)     ;[E121]
004022  132146  000007  000000          RSUB    R7,R6                           ; R6=(GRP/CYL-CUR.GRP)*TRKS/GRP-CUR.TRK
004023                                  ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE
004023  010545  007074  125560          ADD     #SDI.12,UBAR\N,BAR %CALL MULBYT ;[V05] R7=R6*SECS/TRK
004024  131147  000002  010000          SUB     R2,R7                           ; R7=TOTAL SECTORS THIS GROUP   ;[E121]
004025  112141  000007  000000          CMP     R7,R1                           ; IF R7 GTE SECS/REQ            ;[E121]
004026  033467  020001  144027          MOV\T   R1,R7           %TGTE           ;   THEN R7=# SECTORS IN REQUEST;[E121]
004027  013740  007036  125625  20$:    MOV     #SEX2SK,BAR     %CALL   S.STR7  ; SAVE # SECTORS UNTIL SEEK      ;[E121]
004030  010545  007014  135625          ADD     #SDI.SS,UBAR\N,BAR %CALL S.STR7 ; SET # SECTORS B-4 SEEK/IN REQUEST
004031  112547  000041  000000          CMP     #BUFLMT,R7                      ; [16K][U52EC1]IF R7 GR/EQ BUFFER LIMIT
004032  033767  020041  044033          MOV\T   #BUFLMT,R7      %TNCRY          ; [16K][U52EC1]  THEN R7=BUFFER LIMIT
004033  010550  007006  135574          ADD     #P.OPCD,R10\N,BAR %CALL S.LDR1  ; fetch opcode from MSCP packet  ;[E121]
004034  013740  007144  125617          MOV     #OPCODE,BAR     %CALL   S.SLB1  ; and store low byte in OPCODE   ;[E121]
004035  033441  000007  010000          MOV     R7,R1                           ; Store alloc limit in R1 (was in ALOLMT);[E121]

                                ; ***   BUFFER CONTROL BLOCK ALLOCATION LOOP ***

004036  033742  000201  135532          MOV     #NBCB+1,R2      %CALL   S.FBC1  ; [16K]R2=# BUFR CNTRL BLKS/R0=ADDR OF 1ST
004037  013440  007000  010000          MOV     R0,BAR                          ; BAR=R0(ADDR OF 1ST BUFR CNTRL BLK)
004040  031442  000002  000000  U.ALOH: DEC     R2                              ; [16K]IF LAST BUFFER
004041  013702  010003  104062          MOV     (BUF),R2\N      %JZRO   U.ALOL  ;   THEN DONE/EXIT
004042  030560  017015  054040          ADD\T   #BUF.LL,R0,BAR  %JNZRO  U.ALOH  ; [16K]IF AVAILABLE (EQ 0) THEN PROCESS

                                ; ***   UNUSED BUFFER CONTROL BLOCK FOUND ***                                   ;[E121]
                                                                                                                ;[E121]
004043  010540  007006  135623          ADD     #BUF.SD,R0\N,BAR %CALL  S.STUB  ; SAVE PARENT SDI CTRL BLK PTR IN BUFFER;[E121]
                                                                                                                ;[E121]
004044  013740  007256  125572          MOV     #TEMP,BAR       %CALL   S.LDQQ  ; Q = pointer to previous BCB   ;[E121]
004045  033447  013000  014052          MOV     R0,R7,BUF       %JNZRO  10$     ; update BCB ptr, br if not 1st BCB     ;[E121]
                                                                                                                ;[E121]
                                ; ***   FIRST BUFFER CTL BLOCK SPAWNS SOME SPECIAL ACTIONS ***                  ;[E121]
                                                                                                                ;[E121]
004046  010545  007006  135616          ADD     #SDI.UB,UBAR\N,BAR %CALL S.STR0 ; store as beginning of upper chain     ;[E121]
004047  010545  007007  125616          ADD     #SDI.DB,UBAR\N,BAR %CALL S.STR0 ; store as beginning of lower chain     ;[E121]
004050  033746  000060  132375          MOV     #<SUSP!BFRQ),R6 %CALL   U.CLRS  ; get the lower going - we can          ;[E121]
                                                                                ; build the buffer chain before         ;[E121]
                                                                                ; it needs it...                        ;[E121]
004051  033440  000007  104053          MOV     R7,R0           %JMP    20$     ; restore R0 (U.CLRS kills it)          ;[E121]
```

```
                                                                                                                    ;[E121]
004052  000640  007000  125616  10$:    ADD     #BUF.NL,Q,BAR    %CALL   S.STRO  ; not 1st BCB - chain to prev          ;[E121]
004053  013740  007144  135601  20$:    MOV     #OPCODE,BAR      %CALL   S.LDR7  ; get MSCP opcode in R7               ;[E121]
004054                                   ASSUME  OP.WR&6,NE,0                                                        ;[E121]
004054                                   ASSUME  OP.ERS&6,NE,0                                                       ;[E121]
004054                                   ASSUME  OP.RPL&6,NE,0                                                       ;[E121]
004054                                   ASSUME  <OP.RD!OP.ACC!OP.CMP!OP.ONL>&6,EQ,0                                 ;[E121]
004054  014547  000006  010000           BIT     #6,R7                    ; check for disk-reading opcode            ;[E121]
004055  033447  010000  135101           MOV     R0,R7            %CZRO   U.CPRM  ; if so, calculate buf params now     ;[E121]
004056  013740  007036  125601           MOV     #SEX2SK,BAR      %CALL   S.LDR7  ; Get sectors until seek              ;[E121]
004057  031447  000007  000000           DEC     R7                       ; Decrement them                           ;[E121]
004060  021341  003007  000000           DECB    R7\0,R1,BUF              ; and restore - also decrement alloc limit;[E121]
004061  030540  017015  014040           ADD     #BUF.LL,R0,BAR %JNZR0    U.ALOH  ; [16K]IF NEQ 0 THEN CONTINUE       ;[E121]
                                                                                                                    ;[E121]
                                 ; ***   END BUFFER CONTROL BLOCK ALLOCATION LOOP ***                               ;[E121]
                                                                                                                    ;[E121]
                                 ; ***   MAKE RING AND PRESET U.PROC AND D.PROC BUFFER POINTERS ***                 ;[E121]
                                 ;       Note that the values of BUFLMT and NBCB insure we found at least one BCB  ;[E121]

                                 U.ALOL:
004062  010545  007006  135601           ADD     #SDI.UB,UBAR\N,BAR %CALL S.LDR7 ; [EC0#2]R7=PTR TO FIRST BUFFER
004063  013740  007256  135602           MOV     #TEMP,BAR        %CALL   S.LD10  ; [V05][EC0#2]R10=PTR TO LAST ASSIGNED BUF
004064  013740  007036  125573           MOV     #SEX2SK,BAR      %CALL   S.LDR0  ; R0=# SECTORS UNTIL SEEK
004065  133587  010200  144066           BIS\T   #BLAST,R7        %TZRO           ; IF 0 THEN LAST BUF B-4 SEEK
004066  010550  007000  125602           ADD     #BUF.NL,R10\N,BAR %CALL S.LD10  ; R10=LINK WORD
004067  033150  003007  010000           OR      R7,R10,BUF                      ; OR IN PTR TO 1ST BUFFER

                                 ; ***   CHECK IF MAPPING ACTIVE ON THIS COMMAND AND IF SO INIT MAPPING STRUCTURES if necessary

004070  010545  007022  135602           ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [QDA]R10=MSCP PACKET PTR
004071  010550  007013  125574           ADD     #P.BUF1,R10\N,BAR %CALL S.LDR1  ; [QDA]GET HI ORDER FIRST LONGWORD BUF DESC;[E121]
004072                                   ASSUME  MAPVAL,EQ,BIT15                 ; [QDA]                             ;[E121]
004072  013740  037144  004103           MOV     #OPCODE,BAR      %JNMSB U.ALOZ  ; [QDA]IF NO MSB THEN NO MAPPING    ;[E121]
004073  033700  000003  000000           MOV     (BUF),R0                       ; [ch11]GET THE OPCODE              ;[E121]
004074                                   ASSUME  <OP.RPL&OP.ACC&OP.ERS>&20,NE,0  ; [ch11]MAKE SURE CHECK WILL WORK   ;[E121]
004074                                   ASSUME  <OP.RD!OP.WR!OP.CMP!OP.ONL>&20,EQ,0                                 ;[E121]
004074  014540  000020  010000           BIT     #20,R0                         ; [ch11]SEE IF ACC, ERS, OR RPL
004075  016540  010011  004103           XOR     #OP.ONL,R0\N     %JNZR0 U.ALOZ  ; [ch11]SEE IF ONLINE/BYPASS MAP TEST IF
                                                                                ;  ACC, ERS, OR RPL
004076  013440  010000  104103           NOP                      %JZR0  U.ALOZ  ; [ch11]IF ONLINE, BYPASS MAP CHECK TOO
004077  010545  007115  125574           ADD     #MAP.ST,UBAR\N,BAR %CALL S.LDR1 ;[QDA]SEE IF MAPPING INIT NEEDED
004100  013440  010000  014103           NOP                      %JNZR0 U.ALOZ ;[QDA]IF INIT DONE, CONTINUE WITHOUT CALLING INIT
004101  013440  000000  125232           NOP                      %CALL  U.MINT ; [QDA]INIT MAPPING VARS AND READ IN PTE'S
004102  033767  010026  052071           MOV\T   #ER.MRR,R7       %JNZR0 U.UERR ; [QDA]IF NOT ZERO THEN ERROR IN BUS ACCESS [rae4-c119]
                                 U.ALOZ:

                                 ; ***   CHECK FOR WRITE OR ERASE FOR IMMMEDIATE BUFFER FILL ***

004103  010545  007024  125630           ADD     #SDI.OE,UBAR\N,BAR %CALL S.CLRB ; [U52EC1]ZAP OVERLAP ENABLE FLAG
004104  013740  007144  125574           MOV     #OPCODE,BAR      %CALL   S.LDR1  ; R1 = MSCP opcode                  ;[E121]
004105                                   ASSUME  OP.WR&6,NE,0                                                        ;[E121]
004105                                   ASSUME  OP.ERS&6,NE,0                                                       ;[E121]
004105                                   ASSUME  OP.RPL&6,NE,0                                                       ;[E121]
004105                                   ASSUME  <OP.RD!OP.ACC!OP.CMP!OP.ONL>&6,EQ,0                                 ;[E121]
```



```
004105  014541  000006  010000           BIT     #6,R1                           ; check for disk-writing opcode     ;[E121]
004106  036541  010041  054147           XOR\F   #OP.RD,R1        %JNZR0  U.BFSV ; if write/erase/replace, start filling bufs;[E121]
                                                                                ; else R1 = 0 if opcode is READ     ;[E121]
                                                                                                                    ;[E121]
                                 ; ***   MUST BE READ OR ACCESS OR COMPARE COMMAND ***                             ;[E121]
                                 ; ***   OVERLAP IS ONLY POSSIBLE ON SIMPLE READS TO SINGLE-UNIT DEVICES ***       ;[E121]
                                                                                                                    ;[E121]
004107  133740  010340  004117           MOV     #<DRV.U2!DRV.U3!DRV.U4>,R0 %JNZR0 10$ ; Overlap on reads only      ;[E121]
004110  010545  007043  135603           ADD     #SDI.UN,UBAR\N,BAR %CALL S.BITO                                     ;[E121]
004111  033740  010001  014117           MOV     #UF.CMR,R0       %JNZR0  10$    ; No overlap if multi-unit drive    ;[E121]
004112  010545  007060  125603           ADD     #SDI.UF,UBAR\N,BAR %CALL S.BITO                                     ;[E121]
004113  010545  017022  014117           ADD     #SDI.PQ,UBAR\N,BAR %JNZR0 10$   ; No overlap if auto-compare        ;[E121]
004114  033708  000003  000000           MOV     (BUF),R6                                                            ;[E121]
004115  010546  007007  125573           ADD     #P.MOD,R6\N,BAR %CALL   S.LDR0  ; [U52EC1]GET OPCODE MODIFIER       ;[E121]
004116  010545  017024  125631           ADD     #SDI.OE,UBAR\N,BAR %CZR0 INIT1  ; [U52EC1]IF EQ 0 THEN OK TO OVERLAP THIS CMD;[E121]
004117  013740  007036  125573  10$:     MOV     #SEX2SK,BAR      %CALL   S.LDR0  ; [QDA]R0=# SECTORS UNTIL SEEK      ;[E121]
004120  010545  007014  105616           ADD     #SDI.SS,UBAR\N,BAR %JMP S.STRO  ; [16K]STORE REDUCED SECTORS UNTIL SEEK/RETURN;[E121]
                                                                                                                    ;[E121]
004121  033701  000003  010000  CKWROM: MOV     (BUF),R1                        ; [U52EC1]R1=SDI.OM (PTR TO OVERLAPPED CMD)
004122  010545  017022  044124           ADD\F   #SDI.PQ,UBAR\N,BAR %JNZR0 CKWREA; [U52EC1]IF OM NEQ 0 THEN CONTINUE

                                 ; ***   ROUTINE TO CHECK FOR WRITE,ERASE, OR REPLACE COMMAND ***
                                 ; ***   SETS "OPCODE" TO MSCP OPCODE FROM P.OPCD(R10) AS A SIDE EFFECT ***         ;[E121]

004123  033701  000003  010000  CKWRER: MOV     (BUF),R1                        ; R1=PKT PTR
004124  030541  007006  125574  CKWREA: ADD     #P.OPCD,R1,BAR   %CALL   S.LDR1  ; R1=OP CODE
004125  013740  007144  125617           MOV     #OPCODE,BAR      %CALL   S.SLB1  ; SAVE LO BYTE OF OP CODE
004126                                   ASSUME  OP.WR&6,NE,0                                                        ;[E121]
004126                                   ASSUME  OP.ERS&6,NE,0                                                       ;[E121]
004126                                   ASSUME  OP.RPL&6,NE,0                                                       ;[E121]
004126                                   ASSUME  <OP.RD!OP.ACC!OP.CMP!OP.ONL>&6,EQ,0                                 ;[E121]
004126  014541  000006  137777           BIT     #6,R1            %RET           ; Return 0 if not write/erase/replace;[E121]
                                                                                                                    ;[E121]
                                 ;       Divide 32-bit # in R0(hi), R1(lo) by 512 - Quo in R1, rem in R0. Q is scratch;[E121]
                                                                                                                    ;[E121]
004127  105541  000376  010000  DIV512: BIC     #177000,R1,Q                    ;Q has remainder                    ;[E121]
004130  134541  000376  000000           AND     #177000,R1                      ;isolate lo quotient                ;[E121]
004131  073141  000000  134136           BIS\L   R0,R1            %CALL   U.L1R1 ;combine quo & start shifting       ;[E121]
004132  073441  000001  124136           MOV\L   R1,R1            %CALL   U.L1R1 ;rotate R1 left 7 places            ;[E121]
004133  073441  000001  124136           MOV\L   R1,R1            %CALL   U.L1R1                                     ;[E121]
004134  073441  000001  000000           MOV\L   R1,R1                                                               ;[E121]
004135  033240  000000  137777           MOV     Q,R0             %RET           ;Put rem in R0 and exit with R0 CC's;[E121]
                                                                                                                    ;[E121]
004136  073441  000001  127777  U.L1R1: MOV\L   R1,R1            %RET                                               ;[E121]
                                                                                                                    ;[E121]
                                         .PAGE
```


LSCS FORM=QUAD

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.RVCT (UPPER PROC REVECTOR DISPATCH)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       IF REVECTORING IS IN PROGRESS ON THIS SDI AND AN UPPER PROCESSOR ACTION IS
                                ; REQUIRED, DISPATCH TO THAT ACTION ROUTINE
                                ;
                                ; INPUTS:
                                ;       UBAR                    POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;
                                ;;-
                                ;
004137  013740  007254  135574  U.RVCT: MOV     #RVCFLG,BAR     %CALL   S.LDR1  ; [UQA]R1=DISPATCH VECTOR
004140  013740  047232  127777          MOV     #RVCSDI,BAR     %RNLSB          ; IF NOT U.PROC TURN THEN RETURN
004141  016505  000003  000000          XOR     (BUF),UBAR\N                    ; [U52EC2]IF THIS SDI NOT ACTIVE
004142  013740  017250  027777          MOV     #RVECUP,BAR     %RNZRO          ; [UQA]  THEN RETURN
004143  033701  000003  010000          MOV     (BUF),R1                        ; [UQA]  ELSE R1=U.PROC VECTOR
004144  037761  013052  154145          COM\T   #<-V.URVC-1>&377,R1,BUF %TZRO   ; [UQA]IF EQ 0 THEN INIT VECTOR
004145  013740  007250  000000          MOV     #RVECUP,BAR                     ; [UQA]BAR=RVECUP
004146  120341  002001  010000          INCB    R1\0,R1,PAR                     ; OTHERWISE DISPATCH TO ROUTINE & INC VECTOR
                                        .PAGE
```



```
                                        .SBTTL  U.PROC WRITE AND ERASE ROUTINE
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.BFSV (BUFFER SERVICE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL FILL EMPTY BUFFERS FROM THE HOST (WRITE), EMPTY FULL
                                ; BUFFERS TO THE HOST (READ), COMPARE BUFFERS WITH HOST DATA (COMPARE), ZERO
                                ; EMPTY BUFFERS (ERASE), OR PERFORM EDC AND ECC CHECKS ON BUFFERS (ACCESS) AT
                                ; THE REQUEST OF THE DRIVE PROCESSOR.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       UPDATED I/O PARAMETERS
                                ;       EMPTY SECTOR BUFFERS (READ)
                                ;       FULL SECTOR BUFFERS (WRITE/ERASE)
                                ;       COMPARED SECTOR BUFFERS (COMPARE)
                                ;       EDC,ECC CHECKS (ACCESS)
                                ;
                                ;;-
                                ;
000040                          SPURT   =       BIT05                           ; transfer 5 sectors before             ;[E121]
                                                                                ; returning to idle loop for polling    ;[E121]
                                                                                ;                                       ;[E121]
004147  013740  007236  000000  U.BFSV: MOV     #BTCNT,BAR                      ; [U52EC1]BAR=PTR TO SPURT CNTR         ;[E121]
004150  013740  003040  010000          MOV     #SPURT,BUF                      ; [U52EC1]SET UP SPURT CNTR             ;[E121]
004151  010545  007006  135601          ADD     #SDI.UB,UBAR\N,BAR %CALL S.LDR7 ; [U52EC1]R7=BUFFER UPROC SUSPENDED ON
004152  033766  010100  142375          MOV\T   #BFSV,R6        %JZRO   U.CLRS  ; [U52EC1]IF R7=0 THEN CLEAR BFSV/RETURN
004153  013440  000000  131622          NOP             %CALL   U.HTMO  ; [U52EC1]RESET HOST TIMEOUT
004154  010547  007000  125573          ADD     #BUF.NL,R7\N,BAR %CALL  S.LDRO  ; [16K]R0=BUFFER STATUS
004155  010545  007025  134121          ADD     #SDI.OM,UBAR\N,BAR %CALL CKWROM ; [U52EC1]IF NOT WRITE/ERASE OP CODE
004156  114540  010100  114327          BIT     #BFULL,R0       %JZRO   U.BMTY  ; [16K]  THEN MUST BE COMPARE,ACCESS OR READ;[E121]
004157  114540  010200  104165          BIT     #BLAST,R0       %JZRO   U.BFLA  ; fast check - br if buf needs filling  ;[E121]
004160  033746  000040  112375          MOV     #SUSP,R6        %JMP    U.CLRS  ; otherwise clear suspend & return      ;[E121]

                                ; *-------------------------------------------------------------------------*
                                ; *      W  W  W  RRRRRR   I  TTTTTTT  EEEEEEE                              *
                                ; *      W  W  W  R     R  I     T     E                                    *
                                ; *      W  W  W  R     R  I     T     E                                    *
                                ; *      W  W  W  RRRRRR   I     T     EEEEEE                               *
                                ; *      W  W  W  R   R    I     T     E                                    *
                                ; *      W  W  W  R    R   I     T     E                                    *
                                ; *       WWWWW   R     R  I     T     EEEEEEE                              *
                                ; *-------------------------------------------------------------------------*

                                ; ***   FILL SECTOR BUFFERS WITH HOST DATA  (WRITE COMMAND) ***
                                ; ***   SEE IF U.PROC DONE WITH THIS REQUEST ***

004161  010547  007000  125573  U.BFIL: ADD     #BUF.NL,R7\N,BAR %CALL  S.LDRO  ; [16K]R0=BUFFER STATUS
004162  114540  000100  000000          BIT     #BFULL,R0                       ; [16K]TEST IF BUFFER FULL
004163  114540  010200  104165          BIT     #BLAST,R0       %JZRO   U.BFLA  ; [16K]IF BUFFER EMPTY THEN CONTINUE
004164  033746  000040  112375          MOV     #SUSP,R6        %JMP    U.CLRS  ; [16K]  ELSE CLEAR SUSPEND AND RETURN
```

```
004165  010545  017014  025574  U.BFLA: ADD     #SDI.SS,UBAR\N,BAR %CNZRO S.LDR1; [16K][U52EC1]IF NOT LAST THEN CONTINUE
004166  033740  010002  104471          MOV     #SEEK,RO        %JZRO   U.BLST  ; [16K]IF LAST BUFFER THEN DONE          ;[E121]

                                ; ***   CHECK IF SPURT COUNT HAS EXPIRED AND IF SO, GO IDLE ***

004167  013740  007236  134636          MOV     #BTCNT,BAR      %CALL   U.LRR1  ; [U52EC1]SHIFT SPURT CNTR
004170  013440  043001  004337          MOV     R1,BUF          %JLSB   U.BDSM  ; [U52EC1]IF LSB THEN DISMISS

                                ; ***   CHECK IF SEEK STILL IN PROGRESS, IF SO THEN LIMIT FETCHAHEAD                     ;[E121]
                                                                                                                        ;[E121]
004171  010545  007000  125603          ADD     #SDI.ST,UBAR\N,BAR %CALL S.BITO ; [16K]IF SEEK NOT IN PROGRESS          ;[E121]
004172  105544  010300  114175          BIC     #DMODE!PLOCK,RLL,Q %JZRO 10$    ; [16K]  THEN CONTINUE                  ;[E121]
004173  101640  000201  000000          SUB     #BUFP52-<NBUFR*2>+16.,Q         ; otherwise don't allow the last        ;[E121]
004174  013440  030000  127777          NOP                             %RMSB   ; eight free buffers to be scarfed      ;[E121]
                                                                                ; (eight is arbitrary - one is enough   ;[E121]
                                                                                ; to prevent deadlock)                  ;[E121]
                                ; ***   OBTAIN BUFFER TO FILL [16K] ***                                                  ;[E121]
                                ;       Bug fix - old routine U.ALBF could return with screwy cond codes if buffer found;[E121]
                                                                                                                        ;[E121]
004175  025544  007001  000000  10$:    BIC     #1,RLL\0,BAR                    ; if buffer lock clr                    ;[E121]
004176  033703  040003  144175          MOV\F   (BUF),R3        %JNLSB  10$     ; then wait                             ;[E121]
004177  031464  010004  044201          DEC\T   RLL,RLL         %JNZRO  20$     ; if buffer then decr R11 to alloc & unlock;[E121]
004200  130444  000004  127777          INC     RLL,RLL         %RET            ; else incr R11 to unlock & return      ;[E121]
004201  010547  007002  000000  20$:    ADD     #BUF.BP,R7\N,BAR                ; BAR = ptr to bufr addr                ;[E121]
004202  013440  003003  125101          MOV     R3,BUF          %CALL   U.CPRM  ; go calculate params                   ;[E121]
004203  010547  007001  135576          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR3  ; See if new group selected             ;[E121]
004204                                  ASSUME  BGRUP,EQ,BITOO                  ; if so, get a jump on things           ;[E121]
004204  033746  040040  022375          MOV     #SUSP,R6        %CLSB   U.CLRS  ; by unsuspending the lower now         ;[E121]

                                ; ***   ENTERED FROM REVECTORING TO REFILL BUFFER USED FOR RCT READS ***

004205  133743  000001  135014  U.BFLC: MOV     #SECSZ,R3       %CALL   U.BSET  ; [16K]SET R3 IN CASE REPLACE/GO SET UP I/O PARAMETERS

                                ; ***   DETERMINE IF WRITE, ERASE, OR REPLACE COMMAND ***

004206  016542  010042  154241          XOR\F   #OP.WR,R2\N      %JZRO   U.UNED  ;[mjto7] [16K][qda]IF WRITE COMMAND[rae9-c119]
                                                                                ; handle invalid pte (from U.PTEL) if necy;[E121]
004207  033442  010005  104211          MOV     UBAR,R2         %JZRO   U.UNRD  ; [16K]  THEN GO FILL BUFFER
004210  033741  000105  104217          MOV     #EDSEED,R1      %JMP    U.BFLF  ; [16K]  ELSE DO ERASE/REPLACE COMMAND
                                U.UNRD:                                         ; [QDA]GO READ RO WORDS FROM BUS/DO SEG READ;[E121]
004211  053440  000000  135635          ROT\R   RO              %CALL   UNB.RD  ; [QDA]DO READ OF FIRST SEGMENT         ;[E121]
004212  010547  017011  004247          ADD     #BUF.GP,R7\N,BAR %JNZRO U.UNER  ; [QDA]ERROR IF NOT ZERO                ;[E121]
004213  033700  000003  000000          MOV     (BUF),RO                        ; [BDA]GET GROUP FOR SEG 2              ;[E121]
004214  055540  000377  134503          BIC\R   #LOBYT,RO       %CALL   U.CPGC  ; CHECK PAGE XING, SET UP SEG 2 ADR     ;[E121]
004215  053440  010000  035636          MOV\R   RO,RO           %CNZRO  UN.BRC  ; READ NEXT SEG IF WE NEED TO           ;[E121]
004216  033442  010005  014247          MOV     UBAR,R2         %JNZRO  U.UNER  ; [U52EC3][16K]IF NEQ 0 THEN ERROR

                                ; ***   BUFFER FULL, CHECK IF EDC FULLY CALCULATED,STORE IN BUFFER ***
                                ; ***   NOTE - FOR ERASE AND REPLACE COMMANDS U.CKEB ZEROES ENTIRE BUFFER
                                ; ***   AND CALCULATES EDC, BECAUSE UBAR = POINTER TO START OF BUFFER ***

004217  033443  000001  125175  U.BFLF: MOV     R1,R3           %CALL   U.CKED  ; [16K]FINISH EDC CALC IF NEEDED
004220  133741  000020  124320          MOV     #MD.ERR,R1      %CALL   U.TMDA  ; [16K]CHECK FOR FORCE ERROR MODIFIER
004221  037463  010003  054222          COM\T   R3,R3           %TNZRO          ;   THEN 1'S COMPLEMENT EDC
```



```
                                ; ***   CHECK IF CALLED FROM REVECTORING ***

004222  013440  007002  000000          MOV     R2,BAR                          ; [16K]BAR=PTR TO EDC WORD
004223  013440  003003  124323          MOV     R3,BUF          %CALL   U.RVCK  ; [16K]STORE EDC IN BUFFER
004224                                  ASSUME  RVWRIT,EQ,BITOO                 ; MAKE SURE RVWRIT IS LSB
004224  035560  043001  047015          BIC\T   #RVWRIT,RO,BUF  %JLSB   U.RV3B  ; [U52EC2]IF REVECTOR CALL THEN RETURN

                                ; ***   DECREMENT NUMBER OF SECTORS BEFORE SEEK ***

004225  010545  007014  125614          ADD     #SDI.SS,UBAR\N,BAR %CALL S.DECB ; GET AND DECREMENT SECOTRS TIL SEEK     ;[E121]
004226  010547  017000  014234          ADD     #BUF.NL,R7\N,BAR %JNZRO U.BFLH  ; IF NOT ZERO THEN CONTINUE             ;[E121]
004227  050144  000004  010000  10$:    ADD\R   RLL,RLL                         ; Attempt to lock out D proc            ;[E121]
004230  033700  020003  044227          MOV\F   (BUF),RO        %JNCRY  10$     ; loop until success / RO = buffer status;[E121]
004231  133540  003300  010000          BIS     #<BFULL+BLAST>,RO,BUF           ; INDICATE LAST BUFFER BEFORE SEEK

                                ; ***   CLEAR SUSPEND AND BUFFER SERVICE REQUEST AND RETURN ***

004232  010545  007000  135573  U.BFLX: ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDRO ; RO=SDI STATUS (MUTEXED)
004233  033746  000140  102376          MOV     #<SUSP+BFSV>,R6 %JMP    U.CLSX  ; RESET SUSPEND AND BUFFER SERVICE / RETURN

                                ; ***   UPDATE PARAMETERS,CHECK FOR NEXT BUFFER EMPTY ***

004234  050144  000004  010000  U.BFLH: ADD\R   RLL,RLL                         ; Attempt to lock out D proc            ;[E121]
004235  033707  020003  044234          MOV\F   (BUF),R7        %JNCRY  U.BFLH  ; loop until success / R7 = buf link    ;[E121]
004236  113547  003100  134232          BIS     #BFULL,R7\N,BUF %CALL   U.BFLX  ; GO LOCK OUT D.PROC/SET SDI STATUS      ;[E121]
004237  010545  007006  000000          ADD     #SDI.UB,UBAR\N,BAR              ; BAR=PTR TO U.PROC BUFFER INDEX         ;[E121]
004240  115547  003200  104161          BIC     #BLAST,R7\N,BUF %JMP    U.BFIL  ; [16K]SAVE CURRENT PTR/GO TRY TO FILL NEXT BUFFER
                                        .PAGE
```

```
                                ; *-------------------------------------------------------------------*
                                ; *     EEEEEEE  RRRRRR   RRRRRR    00000   RRRRRR                   *
                                ; *     E        R     R  R     R  0     0  R     R                  *
                                ; *     E        R     R  R     R  0     0  R     R                  *
                                ; *     EEEEEE   RRRRRR   RRRRRR   0     0  RRRRRR                   *
                                ; *     E        R   R    R   R    0     0  R   R                    *
                                ; *     E        R    R   R    R   0     0  R    R                   *
                                ; *     EEEEEEE  R     R  R     R   00000   R     R                  *
                                ; *-------------------------------------------------------------------*

                                ; ***   FATAL UNIBUS TRANSFER ERROR - UPF SET IF TIMEOUT/NOT SET IF PARITY ***

                                ;       (it waits up to 310 usecs for drive responses to come back) [mjt08]
004241  033746  000263  135474  U.UNED: MOV     #179.,R6        %CALL   UNLOCK  ; [mjt07] make sure everything is unlock to avoid a possib
004242  013440  010000  114246  1$:     NOP                     %JZRO   U.UNEE  ; [mjt08] if timed out, exit
004243  010545  007000  135573          ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDRO ; [mjt07] get the current status
004244  014540  000010  010000          BIT     #VECT,R0                        ; [mjt07] hang around until VECT is cleared
004245  031446  010006  014242          DEC     R6,R6           %JNZRO  1$      ; [mjt07]  too avoid the drive
                                                                                ; [mjt07]  getting a fault (90 for 60's, a7 for 81's)
004246  033746  000001  114255  U.UNEE: MOV     #FM.BAD,R6      %JMP    U.UNEA  ; [mjt07] make sure r6 has correct errorlog code

004247  037143  000003  125633  U.UNER: XNOR    R3,R3           %CALL   UNB.RS  ; [U52EC3]RESET NPR/CLEAR UNIBUS
004250  010547  007006  135577          ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; [U52EC3]RESTORE UBAR
004251  033746  000001  134324          MOV     #FM.BAD,R6      %CALL   U.DERP  ; [U52EC3][U52EC1]R6=BUS ADDR ERROR LOG CODE
                                                                                ; [U52EC3]GO DO ERROR CLEANUP IN CASE OVERLAP
004252  031443  000003  000000  U.UNEC: DEC     R3,R3                           ; [U52EC3]DECR CNTR-GIVE TIME FOR D.PROC TO
                                                                                ; [U52EC3]FINISH SEEK COMMAND
004253  033743  010151  044252          MOV\F   #<ST.HST+SC.NOM>,R3 %JNZRO U.UNEC ; [U52EC1]LOOP ON R3/R3=MSCP STATUS
004254  033763  060011  054255          MOV\T   #<ST.HST>,R3    %TNUPF          ; [mjt07][U52EC1]IF NOT UPF, THEN SET NO SUBCODE

                                ; ***   FATAL DISK DATA ERROR - R6 HAS ERROR LOG CODE, R3 HAS STATUS ***

004255  010545  007003  125622  U.UNEA: ADD     #SDI.SW,UBAR\N,BAR %CALL S.STR3 ; SAVE FATAL ERROR CODE
004256  133740  000240  124625          MOV     #<RVCT!SLAT>,R0 %CALL   U.WTST  ; GO WAIT FOR D.PROC TO BLOCK
004257  033742  000100  134323          MOV     #BFSV,R2        %CALL   U.RVCK  ; [U52EC1]R0=SDI.ES
004260  015540  003020  125473          BIC     #RVACTV,R0\N,BUF %CALL  P.LOCK  ; [U52EC2]LOCK OUT D.PROC TO PREVENT RVCT RACE
                                                                                ; [U52EC2]ZAP RVACTV
004261  014540  000021  000000          BIT     #<RVWRIT!RVACTV>,R0             ; [U52EC2]IF REVECTOR WRITE OR ACTIVE
004262  133542  010200  054266          BIS\F   #RVCT,R2        %JNZRO  U.UNEB  ; [U52EC2][U52EC1]  THEN CONTINUE
004263  013740  007232  010000          MOV     #RVCSDI,BAR                     ; [U52EC2]BAR=PTR TO ACTIVE REVECTORING SDI CONTROL BLK
004264  016505  000003  000000          XOR     (BUF),UBAR\N                    ; [U52EC2]IF REVECT SDI SAME AS THIS ONE
004265  013760  013000  144266          MOV\T   #0,BUF          %TZRO           ; [U52EC2]  THEN MUST ZAP RVCSDI TOO
004266  010545  007000  135573  U.UNEB: ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDRO ; [U52EC2]RO=SDI STATUS
004267  033540  000244  010000          BIS     #<XCMP+DERR+SUSP>,R0            ; [U52EC1]SET ERROR,TRANSFER COMPLETE
004270  035140  003002  135474          BIC     R2,R0,BUF       %CALL   UNLOCK  ; [U52EC1]CLEAR BITS IN R2/UNLOCK D.PROC
004271  013440  000000  134577          NOP                     %CALL   U.GLBN  ; [U52EC1][ECO#1]GET LBN IN R1 R2
004272                                  ASSUME  S.LBNH,EQ,S.LBNL+1              ; MAKE SURE SEQUENTIAL
004272  000550  007016  133213          ADD     #S.LBNL,R10,Q,BAR %CALL STR1R2  ; [ECO#1]STORE LBN IN S.LBNL/H

                                ; ***   CHECK FOR ERROR LOG TO SEND AND IF ENABLED SET SLAT ***
                                ; ***   R3 HAS STATUS, R6 HAS ERROR LOG CODE ***

004273  016543  000010  000000  U.CLOG: XOR     #<ST.DAT!SC.FER>,R3\N           ; [U52EC1]IF FORCED ERROR
004274  016543  010007  137777          XOR     #ST.CMP,R3\N    %RZRO           ; [U52EC1]  THEN DONE/IF NOT COMPARE ERROR

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```



```
004275  033740  010020  004300          MOV     #CF.THS,R0      %JNZRO  U.CLGA  ;   THEN CONTINUE
004276  010550  007005  125573          ADD     #P.RSO3,R10\N,BAR %CALL S.LDRO  ; RO=COMPARE MODIFIER RETRY WORD
004277  033740  010020  137777          MOV     #CF.THS,R0      %RZRO           ; IF COMPARE THEN RETURN (NO LOGS)
004300  013740  007170  135603  U.CLGA: MOV     #CNTFLG,BAR     %CALL   S.BITO  ; IF LOGS NOT ENABLED
004301  013440  010000  114314          NOP                     %JZRO   U.CLGB  ;   THEN CONTINUE
004302  013440  000000  125476          NOP                     %CALL   U.GMST  ;   ELSE SET SDI.SW/RO=MUTEXED SDI STATUS
004303  014540  000200  010000          BIT     #XCMP,R0                        ; [U52EC1]IF TRANSFER DONE
004304  114540  010040  004306          BIT     #SLAT,R0        %JNZRO  U.CLGD  ; [U52EC1]  THEN USE THIS LOG/ELSE IF LOG QUEUED
004305  133564  010200  054314          BIS\T   #PLOCK,RLL      %JNZRO  U.CLGB  ; [EC04 17-Feb-1984]  THEN FREE D.PROC/DROP LOG
004306  133540  003040  135474  U.CLGD: BIS     #SLAT,R0,BUF    %CALL   UNLOCK  ; [U52EC1]ELSE SET SLAT/FREE D.PROC
004307  010545  007003  125622          ADD     #SDI.SW,UBAR\N,BAR %CALL S.STR3 ; [U52EC1]SAVE STATUS CODE
004310  010545  007010  135624          ADD     #SDI.1T,UBAR\N,BAR %CALL S.STR6 ; [U52EC1]SAVE ERROR LOG CODE
004311                                  ASSUME  BUF.HH,EQ,BUF.HL+1              ; MAKE SURE SEQUENTIAL
004311  000547  007004  123211          ADD     #BUF.HH,R7,Q,BAR %CALL  LDR2R1  ; GET BUF.HH/BUF.HL IN R2/R1
004312                                  ASSUME  SDI.RH,EQ,SDI.RL+1              ; [QDA]MAKE SURE SEQUENTIAL
004312  000545  007031  123213          ADD     #SDI.RL,UBAR,Q,BAR %CALL STR1R2 ; [QDA]SAVE LBN IN SDI.RL/SDI.RH
004313  013440  000000  114315          NOP                     %JMP    U.CLGC  ; CONTINUE
004314  010545  007017  125630  U.CLGB: ADD     #SDI.E1,UBAR\N,BAR %CALL S.CLRB ; ZAP ERROR RECOVRY WORD
004315  013440  000000  124323  U.CLGC: NOP                     %CALL   U.RVCK  ; RELEASE D.PROC/CHK FOR REVECTOR CALL
004316                                  ASSUME  RVWRIT,EQ,BIT00                 ; MAKE SURE RVWRIT IS LSB
004316  035560  043001  047144          BIC\T   #RVWRIT,R0,BUF  %JLSB   U.RVFA  ; IF FROM REVECTORING THEN JUMP BACK
004317  013440  000000  127777          NOP                     %RET            ;   ELSE RETURN

                                ; ***   ROUTINE TO TEST MSCP MODIFIER IN R1, R10 -> MSCP PKT ***

004320  010545  007022  135602  U.TMDA: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; R10=MSCP PKT POINTER
004321  010550  007007  000000  U.TMOD: ADD     #P.MOD,R10\N,BAR                ; BAR=PTR TO MSCP MODIFIERS
004322  014501  000003  127777          BIT     (BUF),R1        %RET            ; TEST (BUF0 WITH BIT IN R1/RETURN

                                ; ***   CHECK IF CALL FROM REVECTORING ***

004323  010545  007026  115573  U.RVCK: ADD     #SDI.ES,UBAR\N,BAR %JMP S.LDRO  ; RO=SDI EXTENDED STATUS

                                ; ***   READ FATAL ERROR CLEANUP ROUTINE ***

004324  010545  007025  125575  U.DERP: ADD     #SDI.OM,UBAR\N,BAR %CALL S.LDR2 ; [U52EC1]R2=POSSIBLE OVERLAPPED CMD
                                                                                ; clear overlap indicator and           ;[E121]
004325  013740  013000  137777          MOV     #0,BUF          %RZRO           ; [U52EC3/EC1]IF NONE THEN RETURN/else  ;[E121]
004326  033450  000002  113230          MOV     R2,R10          %JMP    U.LNKH  ; [U52EC3/EC1]LINK TO HEAD OF QUEUE/RTN ;[E121]

                                ; ***   REPLACE COMMAND BUFFER FILL ROUTINE - R3 HAS # WORDS ***

                                ;U.BFLR:        ADD     #BUF.SD,R7\N,BAR %CALL  S.LDR1  ; RETRIEVE SDI CTL BLK PTR FOR A MINUTE
                                ;       ADD     #SDI.PQ,R1\N,BAR %CALL  S.LDR1  ; SO WE CAN GET MSCP PACKET PTR
                                ;       ASSUME  P.RBN1,EQ,P.RBNO+1              ; MAKE SURE SEQUENTIAL
                                ;       ADD     #P.RBN1,R1,Q,BAR %CALL  LDR2R1  ; AND FETCH THE REPLACEMENT RBN
                                ;       DEC     UBAR,Q                          ; Q=START OF DATA BUFFER-1
                                ;       ADD     R3,UBAR                         ; [16K]UBAR=PTR TO EDC WORD
                                ;       BIS     #RBNCOD,R2                      ; SET RBN HEADER TYPE IN HI ORDER
                                ;U.BFLS:        MOV\F   #EDSEED,R1      %JNZRO  U.BFLT  ; [16K]IF NOT DONE THEN CONTINUE
                                ;       MOV     UBAR,R2         %JMP    U.BFLF  ; [16K]**NOTE** EDC OF ANY REPEATING PATTERN
                                ;                                               ; WITH PERIOD 2**N, N<8, IS START SEED
                                ;U.BFLT:        ADD     #1,Q,BAR        %CALL   STR1R2  ; [16K]INCR Q/SET BAR/GO STORE R1 AND R2
                                ;       SUB     #2,R3           %JMP    U.BFLS  ; [16K]REDUCE BYTE COUNT AND LOOP

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```

```
                                        .PAGE
                                        .SBTTL  U.PROC READ,COMPARE, ACCESS ROUTINE
                                ; *-------------------------------------------------------------------------------*
                                ; *     RRRRRR   EEEEEEE      A     DDDDDD                                       *
                                ; *     R     R  E           A A    D     D                                      *
                                ; *     R     R  E          A   A   D     D                                      *
                                ; *     RRRRRR   EEEEEE    AAAAAAA  D     D                                      *
                                ; *     R   R    E         A     A  D     D                                      *
                                ; *     R    R   E         A     A  D     D                                      *
                                ; *     R     R  EEEEEEE  A     A  DDDDDD                                       *
                                ; *-------------------------------------------------------------------------------*

                                ; ***   EMPTY SECTOR BUFFERS AND SEND TO HOST (READ COMMAND) ***
                                ; ***   CHECK IF D.PROC HAS FILLED THIS BUFFER ***

004327  016541  010011  137777  U.BMTY: XOR     #OP.ONL,R1\N    %RZRO           ; [16K]IF BUFFER EMPTY THEN RETURN      ;[E121]
004330  013440  010000  103563           NOP                     %JZRO   U.ONLR  ; [16K]IF ONLINE THEN SPECIAL CODE     ;[E121]
                                                                                                                        ;[E121]
004331  010547  007000  125576  U.BMTX:  ADD     #BUF.NL,R7\N,BAR %CALL  S.LDR3  ; [16K][U52EC1]R3=BUFFER STATUS
004332  010545  007024  125574           ADD     #SDI.OE,UBAR\N,BAR %CALL S.LDR1 ; [U52EC1]R1=OVERLAP ENABLE FLAG
004333  114543  030100  114637           BIT     #BFULL,R3       %JMSB   U.OVLP  ; [16K][U52EC1]IF OVERLAP THEN DO IT/ELSE CHK IF BFR FULL

                                ; ***   CHECK IF ECC CORRECTION NEEDED ***

004334  133740  010026  127777  U.BMTW:  MOV     #<BECC+BLRWR+BDSNF>,R0 %RZRO    ; [16K]IF NOT FULL THEN RETURN
004335  013740  007236  134636           MOV     #BTCNT,BAR      %CALL   U.LRR1  ; [U52EC1]GO SHIFT SPURT CNTR
004336  013440  043001  104341           MOV     R1,BUF          %JNLSB  U.BMTZ  ; [U52EC1]IF NOT LSB THEN CONTINUE
004337  013440  000000  125476  U.BDSM:  NOP                     %CALL   U.GMST  ; [U52EC1]GET MUTEXED SDI STATUS
004340  033746  000100  113107           MOV     #BFSV,R6        %JMP    U.STSX  ; [U52EC1]SET BUFFER SERVICE/EXIT
004341  033540  000040  010000  U.BMTZ:  BIS     #BRCTS,R0                       ; [EERREC]CHECK FOR RCTS ALSO
004342  010547  007001  125603           ADD     #BUF.ST,R7\N,BAR %CALL  S.BITO  ; [16K]CHECK FOR ECC ERROR
004343  033700  010003  104403           MOV     (BUF),R0        %JZRO   U.BMTC  ; [V05]IF NO ECC ERROR THEN CONTINUE
004344  135540  003020  000000           BIC     #BECC,R0,BUF                    ; [V05]CLEAR BECC IN CASE COMPARE CANNOT OBTAIN
                                                                                 ; [V05]A BUFFER FOR HOST DATA AND MUST RE-ENTER LATER
004345  010545  007047  000000           ADD     #SDI.2T,UBAR\N,BAR              ; [EERREC]SET UP BUF FOR ERROR CODE
004346  114540  000004  000000           BIT     #BDSNF,R0                       ; [EERREC]DID DATA SYNC NOT FOUND?
004347  003760  013150  054362           MOV\T   #<ST.DAT+SC.DSY>,Q,BUF %JNZRO U.BMTK ; [EERREC]IF SO, SET UP PROPER ERROR CODE AND EXIT
004350  114540  000002  000000           BIT     #BLRWR,R0                       ; [EERREC]DID LOSS READ/WRITE READY OCCUR?
004351  003760  013213  044362           MOV\T   #<ST.DRV+SC.RWR>,Q,BUF %JNZRO U.BMTK ; [EERREC] IF SO, SET UP PROPER ERROR CODE AND EXIT
004352  014540  000040  010000           BIT     #BRCTS,R0                       ; [EERREC] WAS RCT SEARCHED?
004353  003760  013110  044362           MOV\T   #<ST.DAT+SC.HDR>,Q,BUF %JNZRO U.BMTK ; [EERREC] IF SO SET UP ERROR CODE AND EXIT
004354  133741  000002  124320           MOV     #MD.SEC,R1      %CALL   U.TMDA  ;    ELSE CHK FOR SUPRESS ERROR CORRECTION
004355  013460  010600  166450           NOP\T           @RUPF   %CZRO   ECCC    ;    ELSE CALL ECC ROUTINE/@RUPF TO DO CORRECTION
004356  010545  017047  154373           ADD\F   #SDI.2T,UBAR\N,BAR %JZRO U.BMTA ; [EERREC]IF EQ 0 THEN CORRECTED
004357  003740  003350  010000           MOV     #<ST.DAT!SC.ECC>,Q,BUF          ; [EERREC]ELSE SET ERROR
004360  010547  007001  135573           ADD     #BUF.ST,R7\N,BAR %CALL  S.LDRO  ;[mjt09a]fetch status
004361  133540  003040  000000           BIS     #BECER,R0,BUF                   ;[mjt09a]indicate that a fatal ecc error occurred
004362  133741  000001  124320  U.BMTK:  MOV     #MD.SER,R1      %CALL   U.TMDA  ; GO CHECK IF RECOVERY SUPPRESSED

                                ; ***   READ ECC ERROR RECOVERY PROCESSING ***

004363  014440  010000  134731           CLR     ,R0\N           %CZRO   U.RREC  ; IF NO SUPPRESS THEN DO READ RECOVERY
004364  013440  010000  027777           NOP                     %RNZRO          ; IF SUCCESSFUL THEN RETURN
004365  010545  007047  125572           ADD     #SDI.2T,UBAR\N,BAR %CALL S.LDQQ ; [EERREC] STORE ERROR INTO SW

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```



```
004366  010545  007003  135613           ADD     #SDI.SW,UBAR\N,BAR %CALL S.STQQ ; [EERREC]
004367  010547  007001  135573           ADD     #BUF.ST,R7\N,BAR %CALL  S.LDRO  ;[mjt09a] fetch buffer status
004370  114540  000040  000000           BIT     #BECER,R0                       ;[mjt09a] [EERREC] WAS IT AN ECC ERROR?
004371  013440  010000  114431           NOP                     %JZRO   U.BMTE  ; [EERREC] IF NOT, BRANCH

                                ; ***   FATAL ECC ERROR, BUT NEED BEST GUESS DATA, R7 -> BUFFER ***

004372  013440  000000  114403           NOP                     %JMP    U.BMTC  ;[mjt09a]indicate ecc go transfer best guest data

                                ;;[mjt09a]      ADD     #BUF.ST,R7\N,BAR %CALL  S.LDRO  ; [16K]R0=BUF.BP (FLAG FOR FATAL ERRORS)
                                ;;[mjt09a]      BIS     #BECER,R0,BUF   %JMP    U.BMTC  ; INDICATE FATAL ECC GO TRANSFER SECTOR

                                ; ***   ECC CORRECTED ! - HOORAY FOR MIKE, CHECK FOR THRESHOLD VIOLATION OR LOGS ***

004373  010545  007035  125574  U.BMTA:  ADD     #SDI.EC,UBAR\N,BAR %CALL S.LDR1 ; R1=SDI ECC THRESHOLD
004374  134541  000377  125504           AND     #HIBYT,R1       %CALL   S.SWB1  ; ISOLATE ECC THRESHOLD/SWAP BYTES
004375  013740  007134  125573           MOV     #ECOUNT,BAR     %CALL   S.LDRO  ; R0:# OF SYMBOLS IN ERROR
004376  112141  000000  010000           RSUB    R0,R1\N                         ; IF LT THRESHOLD
004377  033441  020000  014403           MOV     R0,R1           %JNCRY  U.BMTC  ; [ECO#2]  THEN CONTINUE
004400  003740  000013  135501           MOV     #11.,Q          %CALL   S.RRR1  ; ADJUST # SYMBOLS LEFT 5 BITS
004401  030541  000350  010000           ADD     #<ST.DAT+SC.ECC>,R1             ; [ECO#2]R1=COMPOSITE STATUS
004402  010545  007047  135620           ADD     #SDI.2T,UBAR\N,BAR %CALL S.STR1 ; [ECO#2]SAVE STATUS IN SDI.2T

                                ; ***   I/O OK - CALCULATE I/O PARAMETERS ***

004403  003445  000005  135014  U.BMTC:  MOV     UBAR,Q          %CALL   U.BSET  ; GO SET 1ST PARAMETERS
                                                                                 ; return 0 if error fetching PTE        ;[E121]
                                ; ***   CHECK FOR COMPARE COMMAND ***

004404  016542  010040  154246           XOR\F   #OP.CMP,R2\N    %JZRO   U.UNEE  ;[mjt07] [V05]IF COMPARE COMMAND[rae9-c119]
004405  016542  010020  114667           XOR     #OP.ACC,R2\N    %JZRO   UNB.CM  ; [V05]IF ACCESS COMMAND
004406  033741  010105  104415           MOV     #EDSEED,R1      %JZRO   U.BMTD  ;   THEN SKIP UNIBUS WRT/ELSE MUST BE COMPARE
                                U.UNWR:                                          ; [QDA]GO WRITE THE BUS/DO SEGMENT WRITE;[E121]
004407  053440  000000  126242           ROT\R   R0              %CALL   UNB.WR  ; [QDA]DO WRITE OF FIRST SEGMENT        ;[E121]
004410  010547  017011  004247           ADD     #BUF.GP,R7\N,BAR %JNZRO U.UNER  ; [QDA]ERROR IF NOT ZERO                ;[E121]
004411  033700  000003  000000           MOV     (BUF),R0                        ; [BDA] GET NEXT SEGMENT'S GROUP        ;[E121]
004412  055540  000377  134503           BIC\R   #LOBYT,R0       %CALL   U.CPGC  ; CHECK PG XING, LOAD SEG2 ADDR IF NEEDED;[E121]
004413  053460  010000  076243           MOV\RT  R0,R0           %CNZRO  UN.BWC  ; WRITE NEXT SEG IF NECESSARY            ;[E121]
004414  013440  010000  014247           NOP                     %JNZRO  U.UNER  ; [U52EC3][U52EC1]IF ERROR THEN GO PROCESS

                                ; ***   UNIBUS I/O OK, SECTOR BUFFER SENT TO HOST, CHECK IF EDC FULLY CALCULATED ***

004415  133740  000040  124624  U.BMTD:  MOV     #BECER,R0       %CALL   U.TBEC  ; IF BECER SET IN BUF.BP
004416  033462  010005  144422           MOV\T   UBAR,R2         %JZRO   1$      ;[mjt09a] [16K]  THEN FATAL ECC ERROR
004417  010547  007006  135577           ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ;[mjt09a]reset ubar for error
004420  010545  007047  135573           ADD     #SDI.2T,UBAR\N,BAR %CALL S.LDRO ;[mjt09a]get error code
004421  013440  000000  114570           NOP                     %JMP    U.DERF  ;[mjt09a]go save error code

                                1$:                                              ;[mjt09a]set up temp ubar for u.cked
004422  033443  000401  135175           MOV     R1,R3   @SUPF   %CALL   U.CKED  ; [16K]FINISH EDC IF NEEDED
004423  016503  000003  000000           XOR     (BUF),R3\N                      ; [V05]IF BUFFER EDC EQ CALCULATED EDC
004424  017503  010003  104431           XNOR    (BUF),R3\N      %JZRO   U.BMTE  ;   THEN CONTINUE/IF EQ 1'S COMPLEMENT EDC;[E121]
004425  033740  010010  104573           MOV     #<ST.DAT+SC.FER>,R0 %JZRO U.DERD ;  THEN HARD FORCED ERROR           ;[E121]
004426  003740  000112  104512           MOV     #<ST.CNT+SC.EDC>,Q %JMP U.BMTR  ;   ELSE DO ERROR RECOVERY           ;[E121]

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```

```
                                        ;[E121]
                        ; ***   COMPARE SECTOR BUFFER MATCHES - CHECK FOR FORCED ERROR EDC ***

004427  010547  007006  135577  UNB.CX: ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; [ECO#2]RESTORE UBAR
004430  033740  030010  114573          MOV     #<ST.DAT+SC.FER>,R0 %JMSB U.DERD; [ECO#2]IF FORCED ERROR THEN PROCESS

                        ; ***   CHECK IF BAD BLOCK ERROR LOG NEEDED ON THIS SECTOR ***

004431  010547  007002  125514  U.BMTE: ADD     #BUF.BP,R7\N,BAR %CALL  S.RELF  ; [16K]RELEASE DATA BUFFER               ;[E121]
004432  010545  007047  135573          ADD     #SDI.2T,UBAR\N,BAR %CALL S.LDRO ; [ECO#2]RO=MSCP STATUS
004433  013740  013000  114445          MOV     #0,BUF          %JZRO   U.BMTF  ; [ECO#2]IF NEQ 0 THEN GO SET UP LOG

                        ; ***   MARK BLOCK BAD, SEND LOG, ETC. RO HAS MSCP STATUS ***
                        ; ***   BUT ONLY IF NUMBER OF ECC SYMBOLS IN ERROR EXCEEDED THRESHOLD !! ***

004434  010547  007001  125572          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDQQ  ; [EERREC]Q=STATUS
004435  114640  000006  010000          BIT     #<BDSNF!BLRWR>,Q                ; [EERREC]WAS ERROR EITHER DATA SYNC NOT FOUND OR LOSS REA
004436  014640  010060  044570          BIT\F   #<BRCTS!BRARS>,Q %JNZRO U.DERF  ;[mjto6] [EERREC]IF SO, EXIT/WAS RCT SEARCHED?
004437  013440  010000  014570          NOP                     %JNZRO  U.DERF  ;[mjto6] [EERREC]IF SO, EXIT
004440  010545  007017  125630          ADD     #SDI.E1,UBAR\N,BAR %CALL S.CLRB ; clear error recovery state            ;[E121]
004441  033746  000002  134577          MOV     #FM.DSK,R6      %CALL   U.GLBN  ; [ECO#2][ECO#1]GO GET LBN IN R1,R2
004442  133546  000200  124604          BIS     #LF.SUC,R6      %CALL   U.SLBN  ; [U52EC2][ECO#2][ECO#1]THEN GO REPORT BAD BLOCK
004443  033443  000000  124273          MOV     R0,R3           %CALL   U.CLOG  ; [U52EC1][ECO#2]RESTORE STATUS TO R3/GO CHK FOR LOGS ENAB
004444  013740  007236  135632          MOV     #BTCNT,BAR      %CALL   INITM1  ; Set spurt counter to -1               ;[E121]
                                                                                ; this forces the BMTY loop to be exited;[E121]
                                                                                ; the IDLE loop to be passed thru, and  ;[E121]
                                                                                ; the U.LOGS to be called to send the   ;[E121]
                                                                                ; log packet, before the SDI.E1 error   ;[E121]
                                                                                ; recovery status can be reused if the  ;[E121]
                                                                                ; next buffer also requires recovery.   ;[E121]

                        ; ***   CHECK FOR LAST BUFFER IN REQUEST OR LAST BEFORE SEEK REQUIRED ***

004445  010547  007001  125574  U.BMTF: ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR1  ; [16K]R1=READ RECOVERY FLAGS
004446  010547  007000  125575          ADD     #BUF.NL,R7\N,BAR %CALL  S.LDR2  ; R2=CURRENT STATUS/NEXT LINK
004447                                  ASSUME  BLAST,EQ,BIT15                  ; MAKE SURE BLAST IS MSB

                        ; ***   IF BLAST SET THEN CHECK FOR I/O RETRY ***

004447  114541  030200  014460          BIT     #BRTRY,R1       %JNMSB  U.BMTI  ; IF NOT LAST THEN CONTINUE
004450  114541  010100  114471          BIT     #BLSTB,R1       %JZRO   U.BLST  ; [U52EC1]IF NOT RETRY THEN DONE
004451  135542  013200  054471          BIC\F   #BLAST,R2,BUF   %JNZRO  U.BLST  ; [U52EC1]IF RETRY AND NOT REALLY LAST THEN ZAP BLAST/CONT

                        ;       Experimental code (RL, 8/87) to solve the following problem:                    ;[E121]
                        ;       If the drive gets a data sync timeout (BDSNF) or R/W Rdy dropout (BLRWR)        ;[E121]
                        ;       and the operation is later recovered, the controller hangs. The                 ;[E121]
                        ;       apparent reason is that the lower processor code in D.EERR leaves               ;[E121]
                        ;       SDI.DB pointing at the buffer in error and marks it full; after the             ;[E121]
                        ;       error is corrected we must advance SDI.DB so the lower can proceed              ;[E121]
                        ;       correctly starting with the sector after the one in error.                      ;[E121]
                        ;                                                                                       ;[E121]
                        ;       We distinguish this case by the following features:                             ;[E121]
                        ;       1)      We have just finished a recovery - BRTRY is set but not BLSTB           ;[E121]
                        ;       2)      SDI.UB = SDI.DB so the lower is hung on this buffer                     ;[E121]
```



```
                        ;       3)      BFULL is clear in the successor to SDI.DB, so this is not               ;[E121]
                        ;               a case where the lower ran completely around the ring on                 ;[E121]
                        ;               an ECC error correction attempt.                                        ;[E121]
                                                                                                                ;[E121]
004452  010542  007000  125602          ADD     #BUF.NL,R2\N,BAR %CALL  S.LD10  ; R10 = link word of successor          ;[E121]
004453  010545  007007  010000          ADD     #SDI.DB,UBAR\N,BAR              ; get SDI.DB                            ;[E121]
004454  006507  000003  000000          XOR     (BUF),R7,Q                      ; compare with SDI.UB                   ;[E121]
004455  105640  000300  010000          BIC     #BLAST!BFULL,Q                  ; remove extraneous bits                ;[E121]
004456  114550  010100  004460          BIT     #BFULL,R10      %JNZRO  U.BMTI  ; If equal, see if successor is full    ;[E121]
004457  115562  013300  154460          BIC\T   #<BLAST!BFULL>,R2\N,BUF %TZRO   ; If so, this must be the case - fix it!;[E125]
                                                                                                                ;[E121]
                        ; ***   UPDATE PARAMETERS,CLEAR D.PROC FLAGS,CHECK FOR LAST BUFFER ***

004460  010545  007014  135573  U.BMTI: ADD     #SDI.SS,UBAR\N,BAR %CALL S.LDRO ; R0=SECTORS UNTIL SEEK/END
004461  031440  010000  154465          DEC\F   R0,R0            %JZRO   U.BMTJ  ; IF NEQ 0 THEN DECR/ELSE CONTINUE
004462  013440  003000  000000          MOV     R0,BUF                          ; RESET SECTORS UNTIL SEEK
004463  133562  010200  144464          BIS\T   #BLAST,R2        %TZRO          ; IF NOW 0 THEN SET BLAST
004464  135542  000100  135101          BIC     #BFULL,R2        %CALL  U.CPRM  ; [16K]GO CALCULATE PARAMETERS
004465  010547  007000  135473  U.BMTJ: ADD     #BUF.NL,R7\N,BAR %CALL  P.LOCK  ; [16K]BAR=PTR TO NEXT LINK/LOCK OUT D.PROC
004466  033447  003002  124232          MOV     R2,R7,BUF        %CALL  U.BFLX  ; RESET R7/GO CLR SUSP & BUFFER SERVICE
004467  010545  007006  000000          ADD     #SDI.UB,UBAR\N,BAR              ; BAR=PTR TO U.PROC BUFFER INDEX
004470  115547  003200  114331          BIC     #BLAST,R7\N,BUF %JMP    U.BMTX  ; [16K]SAVE CURRENT PTR/GO TRY TO DUMP NEXT BUFFER

                        ; ***   LAST BUFFER READ/WRITTEN - FINISH UP REQUEST ***

004471  033746  000062  135224  U.BLST: MOV     #<SUSP!BFRQ!SEEK>,R6 %CALL U.CKTC ; [U52EC1]IF TRANSFER COUNT EQ 0
004472  033766  010240  144473  U.XCMA: MOV\T   #<SUSP!XCMP>,R6 %TZRO           ; [U52EC1]  THEN R2=XCMP AND SUSPEND
004473  010545  007025  125602          ADD     #SDI.OM,UBAR\N,BAR %CALL S.LD10 ; [U52EC1]R10=POSSIBLE OVERLAPPED CMD
004474  033766  010240  063230          MOV\T   #<SUSP!XCMP>,R6 %CNZRO  U.LNKH  ; [U52EC1]IF EXISTS THEN LINK TO HEAD OF QUEUE
004475  013440  000000  125476          NOP                     %CALL   U.GMST  ; [U52EC1]GET MUTEXED SDI STATUS IN R0
004476  014540  000022  000000          BIT     #<BFRQ!SEEK>,R0                 ; [U52EC1]IF OVERLAPPED FRAGMENT SEEK
004477  035566  010022  044500          BIC\T   #<BFRQ!SEEK>,R6 %TNZRO          ; [U52EC1]  THEN DON'T RESET SEEK
004500  035540  000100  123103          BIC     #BFSV,R0        %CALL   U.SETX  ; [U52EC1]GO SET BITS/RETURN
004501  014546  000002  124655          BIT     #SEEK,R6        %CALL   U.CROS  ; [U52EC1]GO UPDATE GROUP
004502  013440  000000  115505          NOP                     %JMP    S.RELC  ; [16K][U52EC1]RELEASE ALL BUFR CNTRL BLKS
                                                                                                                ;[E121]
                        ;       Routine to load seg2 address and byte ct if seg2 wd ct is non-zero              ;[E121]
                        ;       Inputs: wd ct in R0<14:7> (BDA) or R0<14:6> (QDA), cond codes reflect R0        ;[E122]
                        ;       R7 -> buf ctl blk, R1 = EDC                                                     ;[E121]
                        ;       Outputs: R0<8:0> = byte ct, (IUAR,SAVUAR)=addr, R1 preserved. R3 scratch        ;[E121]
                                                                                                                ;[E121]
004503  053440  010000  127777  U.CPGC: MOV\R   R0,R0                  %RZRO           ; shift wd ct right, exit if 0  ;[E121]
004504  033443  000001  125223          MOV     R1,R3           %CALL   U.R1RO  ; save EDC temporarily                  ;[E122]
004505  010547  007013  125600          ADD     #BUF.U1,R7\N,BAR %CALL S.LDIU   ; IUAR = low address                    ;[E121]
004506  010547  007014  135536          ADD     #BUF.U2,R7\N,BAR %CALL S.SXB1   ; R1 = SAVUAR = hi addr                 ;[E121]
004507  033441  000003  125223          MOV     R3,R1           %CALL   U.R1RO  ; restore EDC to R1, shift R0           ;[E121]
004510  053440  000000  010000          MOV\R   R0,R0                           ;[BDA] keep shifting                    ;[E121]
004511  053440  000000  105223          MOV\R   R0,R0           %JMP    U.R1RO  ;finish shift, exit                     ;[E121]
                                        .PAGE                                                                   ;[E121]
```

```
                                                                                                                    ;[E121]
                                                                                                                    ;[E121]
                                ; *-------------------------------------------------------------------*
                                ; *     EEEEEEE  RRRRRR   RRRRRR    00000   RRRRRR                    *
                                ; *     E        R     R  R     R  0     0  R     R                   *
                                ; *     E        R     R  R     R  0     0  R     R                   *
                                ; *     EEEEEE   RRRRRR   RRRRRR   0     0  RRRRRR                    *
                                ; *     E        R   R    R   R    0     0  R   R                     *
                                ; *     E        R    R   R    R   0     0  R    R                    *
                                ; *     EEEEEEE  R     R  R     R   00000   R     R                   *
                                ; *-------------------------------------------------------------------*

                                ; ***   EDC ERROR RECOVERY ***

                                U.BMTR:                                                                             ;[E121]
                                                                                                                    ;[E121]
                                ;       Check the LOG and ANTILOG RAMs to verify they have good contents.           ;[E121]
                                ;       Bad contents can produce EDC errors on ECC correction.                      ;[E121]
                                ;       If bad contents found, halt immediately - something is seriously wrong!     ;[E121]
                                                                                                                    ;[E121]
000071                          ALGERR  := 71                                   ; error code to report              ;[E121]
                                                                                                                    ;[E121]
004512  003740  000071  010000          MOV     #ALGERR,TSTCNT                  ; in case of error, set value       ;[E121]
004513  137740  000374  010000          MOV     #1777,R0                        ; initialize x = 1023.              ;[E121]
004514  113540  007070  125574  10$:    BIS     #ALGADR,R0\N,BAR %CALL  S.LDR1  ; R1=ALOG(R0)                       ;[E121]
004515  113541  007074  000000          BIS     #LGADR,R1\N,BAR                 ; BAR ppints to LOG(ALOG(R1))       ;[E121]
004516  016500  000003  000000          XOR     (BUF),R0\N                      ; if x not = LOG(ALOG(x))           ;[E121]
004517  031440  010000  010027          DEC     R0              %JNZRO  ERRB1Z  ; then fatal error/else update x   ;[E121]
004520  013440  010000  004514          NOP                     %JNZRO  10$     ; loop until x=0                    ;[E121]
004521  113740  007070  135574          MOV     #ALGADR,BAR     %CALL   S.LDR1  ;  then test special case           ;[E121]
004522  113541  007074  125574          BIS     #LGADR,R1\N,BAR %CALL   S.LDR1  ;  1777 = LOG( ALOG( 0 ))          ;[E121]
004523  137541  000374  010000          XOR     #1777,R1                                                            ;[E121]
004524  013440  010000  010027          NOP                     %JNZRO  ERRB1Z                                      ;[E121]
004525  010545  007022  135602          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [U52EC1]R10=PKT PTR               ;[E121]
004526  010545  007024  125630          ADD     #SDI.OE,UBAR\N,BAR %CALL S.CLRB ; [U52EC1]DISABLE OVERLAP SEEK
004527  010545  007024  125630          ADD     #SDI.OE,UBAR\N,BAR %CALL S.CLRB ; [U52EC1]ELIMINATE D.PROC WINDOW

                                ; ***   CHECK FOR SEEK IN PROGRESS AND WAIT UNTIL DONE ***

004530  010545  007000  125574  U.BMTT: ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR1 ; [U52EC1]R1=SDI STATUS
004531  014541  000002  000000          BIT     #SEEK,R1                        ; [U52EC1]IF SEEK SET
004532  014541  010020  114535          BIT     #BFRQ,R1                %JZRO   U.BMTU  ; [U52EC1]  THEN CHK BFRQ
004533  010545  007020  125575          ADD     #SDI.EO,UBAR\N,BAR %CALL S.LDR2 ; [U52EC1]R2=SDI ERROR RECOV WORD
004534  014541  030020  014530          BIT     #BFRQ,R1                %JNMSB  U.BMTT  ; [U52EC1]IF NO RECOV THEN LOOP
004535  010545  017003  114542  U.BMTU: ADD     #SDI.SW,UBAR\N,BAR %JZRO U.BMTV ; [U52EC1]IF NOT OVERLAPPED FRAGMENT THEN CONTINUE

                                ; ***   OVERLAPPED SEEK EDC ERROR RECOVERY ***

004536  013740  003000  125505          MOV     #0,BUF                  %CALL   S.RELC  ; [U52EC1]RELEASE ALL BUFFERS,ETC
004537  013440  000000  134324          NOP                             %CALL   U.DERP  ; U.DERP will clear the overlap flag and ;[E121]
                                                                                        ; if overlaping in progress queue the    ;[E121]
                                                                                        ; correct packet to the head of the list ;[E121]
004540  010550  007007  125574          ADD     #P.MOD,R10\N,BAR %CALL  S.LDR1  ; [U52EC1]GET MODIFIER FIELD
004541  133541  003040  103131          BIS     #MD.NOV,R1,BUF  %JMP    U.IOPJ  ; [U52EC1]SET NO OVERLAP MODIFIER/RESTART COMMAND
```



```
                                ; ***   NON-OVERLARPED EDC ERROR RECOVERY ***

004542  013440  000000  134731  U.BMTV: NOP                             %CALL   U.RREC  ; [U52EC1]GO DO READ RECOVERY
004543  013440  010000  002014          NOP                             %JNZRO  U.VAXP  ; [U52EC1]IF OK THEN CHK FOR VAX PURGE
004544  033740  000112  114573          MOV     #<ST.CNT!SC.EDC>,R0 %JMP U.DERD ; [U52EC1]  ELSE FATAL EDC ERROR

                                ; ***   COMPARE COMMAND ERROR RECOVERY ANALYSIS ***

004545  033740  000151  010000  U.BMTQ: MOV     #<ST.HST!SC.NOM>,R0             ; [V05]R0=NON-EXISTENT MEMORY ERROR CODE
004546  033760  060211  054547          MOV\T   #<ST.HST!SC.PAR>,R0 %TNUPF      ; [V05]IF NOT UPF THEN PARITY ERROR
004547  010547  007006  135577          ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; [V05]RESTORE UBAR
004550  010545  007022  135602          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [cho04]RESTORE R10 (PKT PTR)    [rae11]
004551  010550  007005  135600          ADD     #P.RSO3,R10\N,BAR %CALL S.LDR6  ; R6=ORIGINAL OP CODE             [rae11]
004552  136546  013001  154247          XOR\F   #BIT08,R6,BUF   %JZRO   U.UNER  ; IF EQ 0 THEN MUST BE COMPARE COMMAND [rae11]
004553  010545  007022  135602  U.BMTS: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [cho04]RESTORE R10 (PKT PTR)
004554  010550  007005  135600          ADD     #P.RSO3,R10\N,BAR %CALL S.LDR6  ; R6=ORIGINAL OP CODE
004555  136546  013001  154573          XOR\F   #BIT08,R6,BUF   %JZRO   U.DERD  ; IF EQ 0 THEN MUST BE COMPARE COMMAND
004556  010550  007006  135574          ADD     #P.OPCD,R10\N,BAR %CALL S.LDR1  ; [[16K]ECO#2]ELSE R1=CMP OP CODE & FLAGS
004557  114546  000001  000000          BIT     #BIT08,R6                       ; IF 2ND TIME THRU
004560  134541  010340  104573          AND     #<EF.BBR!EF.BBU!EF.LOG>,R1 %JZRO U.DERD ; [16K][ECO#2]  THEN FINISHED/ELSE ISOLATE MSCP FLAG
004561  033141  003006  000000          OR      R6,R1,BUF                       ; [ECO#2]OR IN ORIG OP CODE
004562  033443  000000  010000          MOV     R0,R3                           ; [U52EC1]SAVE STATUS IN R3
004563  133740  000240  124625          MOV     #<RVCT!SLAT>,R0 %CALL   U.WTST  ; GO WAIT FOR BITS TO BE SET
004564  013440  000000  125505          NOP                             %CALL   S.RELC  ; [16K]RELEASE ALL BUFR CNTRL BLKS
004565  033746  000002  000000          MOV     #FM.DSK,R6                      ; [U52EC2][16K]SET R2 TO DISK ERROR LOG
004566  133546  000100  134273          BIS     #LF.CON,R6              %CALL   U.CLOG  ; [U52EC2]GO CHK FOR ERROR LOG
004567  013440  000000  113131          NOP                             %JMP    U.IOPJ  ; RETRY ORIG OPERATION

                                ; ***   FATAL DATA ERROR, JUST GAVE BEST GUESS, R7 -> BUFFER, SDI.SW HAS ERROR STATUS ***

004570  010545  007003  135616  U.DERF: ADD     #SDI.SW,UBAR\N,BAR %CALL S.STRO ;[mjto6][EERREC] must store error status from u.bmte
004571  010547  007006  135577  U.DERC: ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; RESTORE UBAR
004572  010545  007003  135573          ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDRO ; RO=ERROR STATUS

                                ; ***   FATAL DATA TRANSFER ERROR - RO HAS MSCP STATUS, R7 -> BUFFER ***

004573  013440  000000  134324  U.DERD: NOP                             %CALL   U.DERP  ; [U52EC1]GO DO ERROR CLEANUP
004574  013440  000000  134577  U.DERE: NOP                             %CALL   U.GLBN  ; [U52EC1][16K][ECO#1]GET LBN IN R1,R2
004575  033746  000002  124604          MOV     #FM.DSK,R6              %CALL   U.SLBN  ; [U52EC1]GO SET FLAGS/SET BAD LBN,ETC.
004576  033443  000000  104255          MOV     R0,R3                   %JMP    U.UNEA  ; [U52EC1]GO CHECK FOR LOGS ENABLED AND FINISH UP
                                        .PAGE
```

```
                                ; *-------------------------------------------------------------------------*
                                ; *      RRRRRR     00000   U      U  TTTTTTT     III     N      N  EEEEEEE   *
                                ; *      R     R   0     0  U      U     T         I      NN     N  E         *
                                ; *      R     R   0     0  U      U     T         I      N N    N  E         *
                                ; *      RRRRRR    0     0  U      U     T         I      N  N   N  EEEEE     *
                                ; *      R   R     0     0  U      U     T         I      N   N  N  E         *
                                ; *      R    R    0     0  U      U     T         I      N    NN  E         *
                                ; *      R     R    00000    UUUUU      T        III     N     N  EEEEEEE   *
                                ; *-------------------------------------------------------------------------*

                                ; ***   SUBROUTINE TO GET LBN IN R1,R2 ***

004577  010545  007022  135602  U.GLBN: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [ECO#1]RESTORE R10(MSCP PKT PTR)
004600                                  ASSUME  BUF.HH,EQ,BUF.HL+1              ; [ECO#1]MAKE SURE SEQUENTIAL
004600  000547  007004  123211          ADD     #BUF.HH,R7,Q,BAR %CALL  LDR2R1  ; [ECO#1]R2/R1=HI/LO BAD LBN
004601  114542  000360  000000          BIT     #HDCOD,R2                       ; [ECO#1]IF NOT LBN
004602                                  ASSUME  SDI.RH,EQ,SDI.RL+1              ; [QDA][ECO#1]MAKE SURE SEQUENTIAL
004602  000545  017032  033211  U.GLBX: ADD     #SDI.RH,UBAR,Q,BAR %CNZRO LDR2R1 ; [QDA][U52EC2][ECO#1]  THEN R2,R1=HI/LO SAVED LBN
004603  135542  000360  127777          BIC     #HDCOD,R2       %RET            ; [ECO#1]STRIP OFF HEADER CODE/RETURN

                                ; ***   SUBROUTINE TO CHECK FLAGS, PUT BAD LBN IN PKT,ETC., R0=MSCP STATUS ***

004604  010545  007022  135602  U.SLBN: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; R10=MSCP PKT PTR
004605  010550  007006  125576          ADD     #P.FLGS,R10\N,BAR %CALL S.LDR3  ; R3=MSCP FLAGS
004606  016540  000010  000000          XOR     #<ST.DAT!SC.FER>,R0\N           ; [ECO#1]IF FORCED ERROR
004607  016540  010007  114614          XOR     #ST.CMP,R0\N    %JZRO   1$      ; [mjt09][ECO#1]  THEN NOT BAD/IF COMPARE ERROR
004610  016540  010112  137777          XOR     #<ST.CNT!SC.EDC>,R0\N %RZRO     ; [ECO#1]THEN NOT BAD/IF EDC ERROR
004611  114543  010200  114614          BIT     #EF.BBR,R3      %JZRO   1$      ; [mjt09][ECO#1]  THEN NOT BAD BLK/IF BAD BLK REPORTED
004612  133563  013100  067777          BIS\T   #EF.BBU,R3,BUF  %RNZRO          ;   THEN MARK UNREPORTED/RETURN
004613  013440  000000  104622          NOP                     %JMP    2$      ; [mjt09] branch to set bbr

004614  010545  007047  125572  1$:     ADD     #SDI.2T,UBAR\N,BAR %CALL S.LDQQ ; [mjt09a]what was temp error?
004615  010550  017006  127777          ADD     #P.FLGS,R10\N,BAR %RZRO         ; [mjt09]return if zero/reset bar
004616  004640  000037  000000          AND     #ST.MSK,Q                       ; [mjt09a]mask data error
004617  016640  000010  000000          XOR     #<ST.DAT>,Q\N                   ; [mjt09a]did an ecc (data) error occur?
004620  114543  010200  027777          BIT     #EF.BBR,R3      %RNZRO          ; [mjt09][ECO#1]  THEN NOT BAD BLK/IF BAD BLK REPORTED
004621  133563  013100  067777          BIS\T   #EF.BBU,R3,BUF  %RNZRO          ; [mjt09]  THEN MARK UNREPORTED/RETURN
                                2$:

                                ; ***   BAD BLOCK - NOW LET'S DETERMINE WHAT TO REPORT ***

004622  133543  003200  000000          BIS     #EF.BBR,R3,BUF                  ;   [ECO#1]ELSE INDICATE BAD BLK REPORTED
004623                                  ASSUME  SDI.BH,EQ,SDI.BL+1              ; [U52EC1]MAKE SURE SEQUENTIAL
004623  000545  007027  103213          ADD     #SDI.BL,UBAR,Q,BAR %JMP STR1R2  ; [U52EC1]STORE IN SDI CNTRL BLK/RETURN

                                ; ***   ROUTINE TO SEE IF BITS IN R0 ARE SET IN BUF.ST [16K] ***

004624  010547  007001  115603  U.TBEC: ADD     #BUF.ST,R7\N,BAR %JMP   S.BIT0  ; [16K]GO SEE IF R0 BITS SET/RETURN

                                ; ***   ROUTINE TO WAIT FOR BITS IN R0 + SUSP!XCMP TO BE SET IN SDI.ST ***

004625  013740  007156  125630  U.WTST: MOV     #NSEEKS,BAR     %CALL   S.CLRB  ; [ECO#2]PREVENT DEADLOCK,I.E. D.READ/WRIT MUST GET EXECUT
004626  037142  000002  000000          XNOR    R2,R2                           ; [U52EC1]R2=LOOP LIMIT (APPROX 65 MSECS)
004627  006544  010135  137777  U.WTSA: XOR     #<BUFP42-<NBUFR*2>>,RLL,Q %RZRO ; [bda][cho4][U52EC1]

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```



```
                                                                ;Q=XOR OF STK PTR AND NULL BUFFER PTR
004630  004640  000376  000000          AND     #376,Q                          ; [U52EC1]IF NO BUFFERS LEFT (IGNORE LOCK BIT)
004631  031442  010002  114627          DEC     R2              %JZRO   U.WTSA  ; [U52EC1]DECR LOOP COUNTER AND LOOP IF NONE LEFT

                                ; ***   CHECK IF BITS (R0) SET IN SDI STATUS ***

004632  010545  007000  125603          ADD     #SDI.ST,UBAR\N,BAR %CALL S.BIT0 ; IF R0 BITS NOT SET
004633  010565  017025  165572          ADD\T   #SDI.OM,UBAR\N,BAR %CZRO S.LDQQ ; [U52EC1] THEN IF SDI.OM EQ 0
004634  033540  010260  114625          BIS     #<XCMP!SUSP!BFRQ>,R0 %JZRO U.WTST ; [U52EC2][U52EC1] THEN SET SUSP IN R0 & LOOP
004635  013440  000000  127777          NOP                     %RET            ;   ELSE RETURN

                                ; ***   ROUTINE TO LOAD R1 (SHIFTED RIGHT) FROM THE BUFFER ***

004636  053701  000003  137777  U.LRR1: MOV\R   (BUF),R1        %RET            ; [U52EC1]GET R1 SHIFTED RIGHT

                                ; ***   ROUTINE TO PROCESS NEXT COMMAND/FRAGMENT SEEK OVERLAP ***
                                ;       BAR points to SDI.OE(UBAR)                                              ;[E121]
                                                                                                                ;[E121]
004637  013740  003000  135224  U.OVLP: MOV     #0,BUF          %CALL   U.CKTC  ; Clear ovlp flag; if transfer ct = 0   ;[E121]
004640  010545  017022  114644          ADD     #SDI.PQ,UBAR\N,BAR %JZRO U.OVLB ; then try to overlap next command     ;[E121]

                                ; ***   OVERLAP SEEK OF NEXT FRAGMENT ***

004641  033746  000062  123102          MOV     #<SUSP!BFRQ!SEEK>,R6 %CALL U.SETS; [U52EC1]SET SEEK, BFRQ, AND SUSPEND
004642  014546  000002  124655          BIT     #SEEK,R6        %CALL   U.CROS  ; [U52EC1]GO UPDATE GROUP
004643  114543  000100  114334  U.OVLA: BIT     #BFULL,R3       %JMP    U.BMTW  ; [U52EC1]CONTINUE

                                ; ***   CHECK FOR SEEK OVERLAP OF NEXT COMMAND ***

                                U.OVLB:
                                ;       The following code was deleted by RL, 7/31/87, because it                ;[E121]
                                ;       doesn't do anything except inhibit all overlap (Since we got             ;[E121]
                                ;       here from U.BMTX, P.OPCD cannot be OP.ERS!)                               ;[E121]
                                ;                                                                                ;[E121]
                                ;       ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LDR6 ; [mjt07]r6=current cmd ptr      ;[E121]
                                ;       ADD     #P.OPCD,R6\N,BAR %CALL  S.LDR2  ; [mjt07]get opcode              ;[E121]
                                ;       XOR     #OP.ERS,R2\N                    ; [mjt07]is it an erase?         ;[E121]
                                ;       NOP                     %JNZRO  U.OVLA  ; [mjt07] if it is, exit         ;[E121]
                                ;                                                                                ;[E121]
004644  033706  000003  000000          MOV     (BUF),R6                        ; [U52EC1]R6=CURRENT CMD PTR      ;[E121]
004645  013440  007006  125575          MOV     R6,BAR          %CALL   S.LDR2  ; [U52EC1]R2=LINK WORD FOR THAT CMD
004646  105542  000300  000000          BIC     #<PSTAT!PACTV>,R2,Q             ; [U52EC1]ZAP CONTROL BITS
004647  010562  017006  075572          ADD\T   #P.OPCD,R2\N,BAR %CNZRO S.LDQQ  ; [U52EC1]IF PTR NEQ 0 THEN LOAD IT
004650  014640  000040  010000          BIT     #<OP.RD&OP.CMP&OP.WR>,Q         ; [U52EC1]IF NEXT CMD NEQ RD,WRT, OR CMP
004651  013440  010000  104643          NOP                     %JZRO   U.OVLA  ; [U52EC1]  THEN GO BACK & CONTINUE

                                ; ***    OVERLAP SEEK OF NEXT READ, WRITE, OR COMPARE COMMAND ***

004652  010545  007025  135624          ADD     #SDI.OM,UBAR\N,BAR %CALL S.STR6 ; [U52EC1]  ELSE SET SDI.OM
004653  010545  007022  125621          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.STR2 ; [U52EC1]PUT NEXT CMD ON HEAD OF QUEUE
004654  013440  000000  102101          NOP                     %JMP    U.SEKO  ; [U52EC1]GO SELECT NEXT CMD & START SEEK

                                ; ***   ROUTINE FOR CYLINDER CROSSING CYLINDER AND GROUP UPDATE ***

KDBUP  KDB50.MICROCODE.,22-APR-1988 11:16:48.97
```

```
004655  013440  010000  137777  U.CROS: NOP                     %RZRO           ; [U52EC1]IF NO SEEK THEN RETURN
004656  000545  007012  123211          ADD     #SDI.CH,UBAR,Q,BAR %CALL LDR2R1 ; [U52EC1]R1,R2=LO,HI CYL NUMBER
004657  130441  000001  135620          INC     R1,R1           %CALL   S.STR1  ; [U52EC1]INCR LO CYL/RESET
004660  130442  010002  044662          INC\F   R2,R2           %JNZRO  U.CRSA  ; [U52EC1]IF NEQ 0 THEN INCR HI CYL
004661  010545  007012  125621          ADD     #SDI.CH,UBAR\N,BAR %CALL S.STR2 ; [U52EC1]  ELSE BUMP HI CYL/RESET
004662  010545  007022  135602  U.CRSA: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; [U52EC1]R10=MSCP PKT PTR
004663  010550  007026  125630          ADD     #S.GRUP,R10\N,BAR %CALL S.CLRB  ; [U52EC1]RESET GROUP VALUE

                                ; ***   GROUP UPDATE ROUTINE ***

004664  010550  007026  135573  U.SGRP: ADD     #S.GRUP,R10\N,BAR %CALL S.LDRO  ; [U52EC1]R0=GROUP NUMBER THIS CMD
004665  010545  007013  125616          ADD     #SDI.GP,UBAR\N,BAR %CALL S.STRO ; [U52EC1]SAVE AS MASTER GROUP NUMBER
004666  010545  007005  105616          ADD     #SDI.UG,UBAR\N,BAR %JMP S.STRO  ; [U52EC1]SAVE AS U.CPRM GROUP/RETURN
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------------*
                                ; *      CCCCCC   00000   M      M  PPPPPP      A     RRRRRR   EEEEEEE          *
                                ; *     C        0     0  M      M  P     P    A A    R     R  E                *
                                ; *     C        0     0  MM    MM  P     P   A   A   R     R  E                *
                                ; *     C        0     0  M MMM M  PPPPPP   AAAAAAA  RRRRRR   EEEEEE           *
                                ; *     C        0     0  M M M M  P        A     A  R    R    E                *
                                ; *     C        0     0  M  M  M  P        A     A  R     R   E                *
                                ; *      CCCCCC   00000   M  M  M  P        A     A  R     R  EEEEEEE          *
                                ; *-------------------------------------------------------------------------------*
                                ;;+
                                ; ROUTINE NAME:
                                ;       UNB.CM (UNIBUS COMPARE SUBROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE ACQUIRES A BUFFER THEN FILLS IT WITH BUF.BC
                                ; WORDS FROM THE HOST. THE DATA FROM THE HOST (CMPBUF) IS THEN COMPARED WITH
                                ; THE DATA JUST READ FROM THE DISK (BUF.BP). IF ALL BUF.BC WORDS MATCH THEN
                                ; THE EDC'S ARE CHECKED.
                                ;
                                ; INPUTS:
                                ;       IUAR,UAR        STARTING UNIBUS ADDRESS REGISTER IMAGE
                                ;       R7              BUFFER CONTROL AREA POINTER
                                ;
                                ; OUTPUTS:
                                ;       SUCCESS - MERGE BACK INTO READ CODE
                                ;       FAILURE - ENTER U.BMTS (COMPARE FAILURE) OR U.BMTO (I/O FAILURE)
                                ;
                                ;;-

                                ; ***   COMPARE SECTOR BUFFERS WITH HOST DATA (COMPARE COMMAND) ***

004667  133740  000040  124624  UNB.CM: MOV     #BECER,R0       %CALL   U.TBEC  ; GO SEE IF FATAL ECC ERROR
004670  033760  010350  054571          MOV\T   #<ST.DAT+SC.ECC>,R0 %JNZRO U.DERC ; IF FATAL ECC ERROR THEN EXIT
004671  033743  000105  000000          MOV     #EDSEED,R3                      ; MOVE SEED INTO R3
004672  033442  007005  010000          MOV     UBAR,R2,BAR                     ; BAR,R2=BUFFER POINTER
004673  133740  000001  135220          MOV     #SECSZ,R0       %CALL   U.CKEB  ; FINISH EDC IF NEEDED
004674  016503  000003  000000          XOR     (BUF),R3\N                      ; IF BUFFER EDC EQ CALCULATED EDC
004675  017503  010003  104701          XNOR    (BUF),R3\N      %JZRO   UNB.CA  ; THEN CONTINUE/IF NEQ 1'S SOMPLEMENT EDC
004676  003760  010112  044512          MOV\T   #<ST.CNT+SC.EDC>,Q %JNZRO U.BMTR;   THEN DO ERROR RECOVERY              ;[E121]
004677  010547  007006  135572          ADD     #BUF.SD,R7\N,BAR %CALL  S.LDQQ  ;   ELSE RESTORE Q/SET BIT15 OF SDI POINTER
004700  103640  003200  000000          BIS     #BIT15,Q,BUF                    ;   TO INDICATE FORCED ERROR

                                ; ***   ACQUIRE BUFFER FOR HOST DATA RE-READ *** [V05]

004701  010547  007012  125573  UNB.CA: ADD     #BUF.BC,R7\N,BAR %CALL  S.LDRO  ; [V05]R0=BYTE COUNT
004702  053440  000000  010000          ROT\R   R0                              ; [ch04] CONVERT BYTE COUNT TO WORD COUNT
004703  013740  007242  125577          MOV     #CMPBUF,BAR     %CALL   S.LDUB  ; [ch04] POINTER TO THE DEDICATED COMPARE BUFFER PO

                                ; ***   PREPARE FOR AND PERFORM HOST DATA RE-READ ***
                                ; ***   UNIBUS READ SEGMENT ROUTINE

004704  013440  000000  125635  UNB.SR: NOP                     %CALL   UNB.RD  ; [QDA]DO READ OF FIRST SEGMENT
004705  013440  010000  044545          NOP\F                   %JNZRO  U.BMTO  ; [QDA]ERROR IF NOT ZERO
004706  010547  007011  125573          ADD     #BUF.GP,R7\N,BAR %CALL  S.LDRO  ; [BDA]GET SEG 2 GROUP                  ;[E121]
```

```
004707  055540  000377  134503          BIC\R   #LOBYT,R0       %CALL   U.CPGC  ; CHECK PG XING, LOAD SEG2 PARAMS IF NEC;[E121]
004710  013740  007244  135616          MOV     #MAPSAV,BAR     %CALL   S.STRO  ; [cho2]Save SEG 2 byte count for later use;[E121]

                                ; ***   NOTE THAT UBAR IS STILL SET UP FROM THE PREVIOUS CALL    ***  [cho4]
                                ; ***   TO UNB.RD AND POINTS TO THE CORRECT OFFSET WITHIN THE    ***  [cho4]
                                ; ***   DEDICATED COMPARE BUFFER FOR THE SEGEMENT 2 DATA         ***  [cho4]

004711  053440  010000  035636          MOV\R   R0,R0            %CNZRO  UN.BRC  ; [QDA]Else read SEG 2                    ;[E121]
004712  013440  010000  014545          NOP                      %JNZRO  U.BMTO  ; [V05]IF ERROR THEN GO PROCESS

                                ; ***   PERFORM COMPARE FUNCTION (R3 = EDC OF HOST DATA BUFFER) ***

004713  010547  007012  125573          ADD     #BUF.BC,R7\N,BAR %CALL  S.LDR0  ; [V05]R0=BYTE COUNT
004714  013740  007244  125574          MOV     #MAPSAV,BAR      %CALL  S.LDR1  ; [cho2]Restore SEG 2 byte count
004715  030140  000001  000000          ADD     R1,R0                           ; [cho2]Add SEG 2 byte count to SEG 1
004716  010547  007002  125574          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR1  ; [V05]R1=DISK DATA BUF PTR
004717  013740  007242  135575          MOV     #CMPBUF,BAR      %CALL  S.LDR2  ; [V05]R2=HOST DATA BUF PTR
004720  120341  007001  135572  UNB.CB: INCB    R1\0,R1,BAR      %CALL  S.LDQQ  ; [V05]Q=WORD FROM DISK BUFFER
004721  120342  007002  010000          INCB    R2\0,R2,BAR                     ; [V05]BAR=PTR TO WORD FROM HOST BUFFER
004722  006600  000003  010000          XOR     (BUF),Q                         ; [V05]COMPARE DATA FROM HOST AND DISK
004723  131540  010002  014726          SUB     #2,R0            %JNZRO  UNB.CC ; [V05]IF ERROR THEN PROCESS ELSE DECR CNTR
004724  013440  010000  004720          NOP                      %JNZRO  UNB.CB ; [V05]IF NEQ 0 THEN LOOP ELSE BAR=PTR TO HOST BUFR

                                ; ***   DATA FROM HOST MATCHES DATA FROM DISK - NOW CHECK EDC'S ***

004725  013440  000000  114427          NOP                      %JMP    UNB.CX ; [V05] GO CHECK FOR FORCED ERROR

                                ; ***   SECTOR BUFFER HAS COMPARE FAILURE - PERFORM COMPARE RECOVERY AS APPROPRIATE ***

004726  033740  000007  000000  UNB.CC: MOV     #<ST.CMP>,R0                    ; [V05]R0=COMPARE ERROR CODE
004727  010547  007006  135577          ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; [V05]RESTORE UBAR
004730  013440  000000  104553          NOP                      %JMP    U.BMTS ; [V05]GO PROCESS ERROR
                        .PAGE
```



```
                                .SBTTL  READ RECOVERY ROUTINE
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.RREC (READ RECOVERY ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL INITIATE READ RETRY AND RECOVERY (ERROR RECOVERY
                                ; LEVEL 1) FOR SECTORS READ IN ERROR. THE FOLLOWING SEQUENCE IS FOLLOWED:
                                ;       1. IF LEVEL 1 ERROR RECOVERY IS NOT ACTIVE THEN THE FIRST RETRY
                                ;          WILL PERFORM A REAL TIME GROUP SELECT.
                                ;       2. ERROR RECOVERY LEVEL 1 WILL BE ACTIVATED AND A DRIVE SPECIFIED
                                ;          NUMBER OF RETRIES WILL BE PERFORMED BEFORE THE SDI ERROR RECOVERY
                                ;          COMMAND IS USED.
                                ;       3. IF 2) ABOVE FAILS THEN A DRIVE SPECIFIED NUMBER OF RETRIES WILL
                                ;          BE PERFORMED AT EACH ERROR RECOVERY LEVEL THE DRIVE SUPPORTS.
                                ;       4. IF THE READ SUCCEEDS AT ONE OF THE ERROR RECOVERY LEVELS THE
                                ;          DRIVE IS LEFT AT THAT RECOVERY LEVEL FOR THE NEXT OPERATION.
                                ;       5. IF THE READ CONTINUES TO FAIL AFTER ALL ERROR RECOVERY LEVELS
                                ;          ARE EXHAUSTED THEN THE ROUTINE WILL RETURN A HARD ERROR.
                                ;
                                ; INPUTS:
                                ;       Q               MSCP ERROR CODE OF RETRY CAUSING ERROR
                                ;                       SETTING Q IS OPTIONAL
                                ;       R7              POINTER TO SECTOR BUFFER IN ERROR
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK FOR DRIVE
                                ;
                                ; OUTPUTS:
                                ;       ACTIVATED LEVEL 1 ERROR RECOVERY
                                ;       CONDITION CODE  ZERO    - HARD FAILURE
                                ;       CONDITION CODE NONZERO  - ERROR RECOVERY/RETRY INITIATED
                                ;
                                ;;-
004731  010545  007047  135613  U.RREC: ADD     #SDI.2T,UBAR\N,BAR %CALL S.STQQ ; SAVE ERROR FOR SUCCESS ERROR LOG

                                ; ***   MAKE SURE D.PROC IS REALLY BLOCKED ***

004732  010545  007000  135573  U.RRCA: ADD     #SDI.ST,UBAR\N,BAR %CALL S.LDR0 ; R0=SDI STATUS
004733  114540  000240  010000          BIT     #<SLAT+RVCT>,R0                 ; IF SLAT OR RVCT SET
004734  014540  010244  004741          BIT     #<DERR+XCMP+SUSP>,R0 %JNZRO U.RRCX ; [16K]IF SUSPEND SET
004735  013440  010000  014741          NOP                      %JNZRO  U.RRCX ; [16K]  THEN CONTINUE

                                ; ***   IF NO BUFFERS LEFT THEN TREAT AS SUSPEND ***

004736  025544  007001  000000  10$:    BIC     #1,RLL\0,BAR                    ; [16K][cho3]IF BUFFER LOCK CLR
004737  003700  040003  154736          MOV\F   (BUF),Q          %JNLSB  10$    ; [16K][cho3]  THEN WAIT FOR IT
004740  130444  010004  004732          INC     RLL,RLL          %JNZRO  U.RRCA ; [16K]IF BUFFERS THEN LOOP
004741  014540  000204  000000  U.RRCX: BIT     #<DERR+XCMP>,R0                 ; [16K]  ELSE CHECK FOR ERRORS
004742  034460  010000  067777          CLR\T   R0               %RNZRO         ; IF ERROR THEN RETURN

                                ; ***   CHECK IF LEVEL 1 ERROR RECOVERY ACTIVE ***

004743  010545  007035  135604          ADD     #SDI.ER,UBAR\N,BAR %CALL S.LLB0 ; R0=ERROR RECOVERY LEVELS              ;[E121]
004744  010545  007017  125572          ADD     #SDI.E1,UBAR\N,BAR %CALL S.LDQQ ; Q=LEVEL 1 ERROR STATE
```

```
004745                                  ASSUME  ERRINP,EQ,BIT15                   ; MAKE SURE ERRINP IS MSB
004745  100440  030000  154754          INC\F   R0,Q            %JMSB   U.RRCC  ; IF ERROR RECOVERY ACTIVE THEN CONTINUE
004746  103640  000201  010000          BIS     #<ERRINP+256.>,Q                ;   ELSE ACTIVATE ERROR RECOVERY
004747  010547  007001  125574          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR1  ; [16K]R1=CURRENT BUFFER STATUS WD
004750  035541  000002  000000          BIC     #BCGRP,R1                       ; [16K]CLEAR GROUP DONE FLAG
004751  135541  000001  010000          BIC     #BERDN,R1                       ; [EERREC] CLEAR ERROR RECOVERY CMD DONE FLAG
004752  013440  000000  000000          TST     R0                              ; ISOLATE ERROR RECOVERY LEVELS
004753  033561  013001  044754          BIS\T   #BGRUP,R1,BUF   %TNZRO          ; DO GROUP SELECT ON FIRST RETRY (TO RECENTER HEAD)
004754  154640  000017  000000  U.RRCC: AND\R   #RETCNT,Q,R0                     ; ISOLATE RETRY COUNT

                                ; ***   ISOLATE DRIVE'S RETRY COUNT ***

004755  010545  007034  125575          ADD     #SDI.RC,UBAR\N,BAR %CALL S.LDR2 ; R2=DRIVE RETRY COUNT (HI NIBBLE)
004756  074542  000360  010000          AND\L   #HINIB,R2                       ; ISOLATE RETRY COUNT
004757  073762  010100  144760          MOV\TL  #ERRRTC*16.,R2  %TZRO           ; IF EQ 0 THEN USE DEFAULT
004760  070142  000002  000000          ADD\L   R2,R2                           ; R2=R2*4
004761  016140  000002  010000          XOR     R2,R0\N                         ; IF NEQ MAX
004762  014640  010377  004767          BIT     #LOBYT,Q        %JNZRO  U.RRCE  ;   THEN CONTINUE/ELSE TEST RECOVERY LEVEL

                                ; ***   MAXIMUM RETRY COUNT EXCEEDED, DECREMENT ERROR RECOVERY LEVEL ***

004763  001240  010000  167777          DEC\F   Q               %RZRO           ; IF RECOVERY LEVEL EQ 0 THEN HARD ERROR

                                ; ***   CLEAR RETRIES AND INDICATE POTENTIAL OFF TRACK CONDITION ***

004764  105640  000017  010000          BIC     #RETCNT,Q                       ; ZAP RETRY COUNT
004765  010545  007026  125573          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDR0 ; R0=EXTENDED STATUS WORD
004766  033540  003002  010000          BIS     #OFFTRK,R0,BUF                  ; INDICATE POTENTIAL OFF TRACK CONDITION

                                ; ***   UPDATE RETRY COUNT ***

004767  010545  007017  000000  U.RRCE: ADD     #SDI.E1,UBAR\N,BAR              ; BAR=PTR TO LEVEL 1 RECOVERY WORD
004770  100640  000377  125613          ADDC    #377,Q          %CALL   S.STQQ  ; ADD 1 TO RETRY COUNT/RESET

                                ; ***   CHECK FOR GROUP SELECT NEEDED ***

004771  010547  007011  135574          ADD     #BUF.GP,R7\N,BAR %CALL  S.LDR1  ;[BDA] R1=GROUP NUMBER/EXT UNIBUS ADDR BITS
004772  034541  000377  000000          AND     #LOBYT,R1                       ;[BDA] STRIP OFF S2 WORD COUNT
004773  010545  007005  125617          ADD     #SDI.UG,UBAR\N,BAR %CALL S.SLB1 ; SAVE AS CURRENT GROUP FOR U.PROC
004774  010545  007013  125575          ADD     #SDI.GP,UBAR\N,BAR %CALL S.LDR2 ; R2=CURRENT GROUP

                                ; ***   RESET BUFFER STATUS TO EMPTY ***

004775  103740  000200  010000          MOV     #BRTRY,Q                        ; Q=READ RECOVERY FLAG FOR BUF.BP
004776  010547  007000  125573          ADD     #BUF.NL,R7\N,BAR %CALL  S.LDR0  ; R0=BUFFER STATUS
004777                                  ASSUME  BLAST,EQ,BIT15                  ; MAKE SURE BLAST IS MSB
004777  103660  030100  155000          BIS\T   #BLSTB,Q        %TMSB           ; IF REALLY LAST ONE THEN SET REALLY LAST FLAG
005000  036141  000002  010000          XOR     R2,R1                           ; IF GROUPS NEQ
005001  003660  010001  055002          BIS\T   #BGRUP,Q        %TNZRO          ;   [16K]THEN SET GROUP FLAG IN BUF.ST
005002  133540  000200  000000          BIS     #BLAST,R0                       ; TEMPORARILLY MAKE THIS BUFFER LAST SO
                                                                                ; D.READ WON'T UPDATE SDI.DB FOR THIS SECTOR!
005003  135540  003100  000000          BIC     #BFULL,R0,BUF                   ; [16K]INDICATE BUFFER EMPTY

                                ; ***   SET READ RECOVERY FLAGS IF NOT ALREADY SET ***
```



```
005004  010547  007001  135573          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR0  ; [16K]R0=BUFFER POINTER/RECOVERY FLAGS
005005                                  ASSUME  BRTRY,EQ,BIT15                  ; [U52EC2]MAKE SURE BRTRY IS MSB
005005  004660  030001  155006          AND\T   #BGRUP,Q        %TMSB           ; [U52EC2]IF ALREADY SET THEN ISOLATE BGRUP ONLY
005006  135540  000077  010000          BIC     #<BECC!BGOOD!BDSNF!BLRWR!BERDN!BECER>,R0 ;[mjt09a][EERREC]
                                                                                ;[U52EC2][ECO#2]CLEAR POS ERROR FLAG/ECC ERR FLG
005007  033040  003000  000000          BIS     Q,R0,BUF                        ; [16K]SET BUF.ST FLAGS

                                ; ***   RESET SDI STATUS TO ALLOW I/O TO OCCUR AGAIN ***

005010  033746  000040  123371          MOV     #SUSP,R6        %CALL   U.ZPSW  ; R6=SUSP/GO ZAP SDI.SW
005011  133546  000100  125476          BIS     #ERRIP,R6       %CALL   U.GMST  ; R0=MUTEXED SDI STATUS
005012  135540  000040  000000          BIC     #SLAT,R0                        ; THROW ERROR LOG AWAY !
005013  035540  000304  103107          BIC     #<DERR!XCMP!BFSV>,R0 %JMP U.STSX; [ECO#2]ZAP DRV ERR,TRANSFER COMPL,BUFFER SERVICE
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.BSET (BUFFER SERVICE SET UP)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE SETS THE FOLLOWING PARAMETERS FOR PREPARATION OF
                                ; A READ,WRITE,COMPARE OPERATION:
                                ;       R0 = NUMBER OF BYTES TO TRANSFER
                                ;       IUAR = LO HOST TRANSFER ADDRESS
                                ;       EXTENDED ADDRESS BITS IN CR AND CRI
                                ;       UBAR = SECTOR DATA START ADDRESS
                                ;
                                ; INPUTS:
                                ;       UBAR    POINTER TO SDI CONTROL BLOCK
                                ;       R7      POINTER TO CURRENT SECTOR BUFFER
                                ;
                                ; OUTPUTS:
                                ;       AS MENTIONED ABOVE
                                ;
                                ;;-
                                U.BSET:
005014  010545  007025  125602          ADD     #SDI.OM,UBAR\N,BAR      %CALL   S.LD10  ; ARE OVERLAPPED SEEKS IN PROGRESS? ;[E125]
005015  010545  017022  125602          ADD     #SDI.PQ,UBAR\N,BAR      %CZRO   S.LD10  ; [QDA]R10=MSCP PACKET          ;[E125]
005016  013740  007144  135573          MOV     #OPCODE,BAR             %CALL   S.LDRO  ; [ch11]GET THE OPCODE          ;[E121]
005017                                  ASSUME  <OP.RPL&OP.ACC&OP.ERS>&20,NE,0          ; [ch11]MAKE SURE CHECK WILL WORK;[E121]
005017                                  ASSUME  <OP.RD!OP.WR!OP.CMP!OP.ONL>&20,EQ,0                             ;[E121]
005017  014540  000020  010000          BIT     #20,R0                                  ; [ch11]TEST IF ACC, ERS, OR RPL
005020  016540  010011  015075          XOR     #OP.ONL,R0\N            %JNZRO  U.CLRM  ; [ch11+]SEE IF ONLINE/BYPASS MAPPING CHECK
                                                                                        ;  IF ACC, ERS OR RPL
005021  010547  017014  105076          ADD     #BUF.U2,R7\N,BAR        %JZRO   U.CLM1  ; [ch11+]IF ONL, BYPASS MAPPING CHECK ALSO;[E121]
                                                                                        ; make sure no mapping retained ;[E121]
005022  010550  007013  125574          ADD     #P.BUF1,R10\N,BAR       %CALL   S.LDR1  ; [QDA]R1=HI ORD FIRST LONGWORD BUF DESC;[E121]
005023                                  ASSUME  MAPVAL,EQ,BIT15                         ;[E121]
005023  010547  037014  005076          ADD     #BUF.U2,R7\N,BAR        %JNMSB  U.CLM1  ; [QDA][KJK]NO MSB - NO MAPPING ;[E121]
005024  010547  007001  125574          ADD     #BUF.ST,R7\N,BAR        %CALL   S.LDR1  ; [QDA]GET BUFFER STATUS
005025  014541  000010  000000          BIT     #BMAPDN,R1                              ; [QDA] SEE IF MAPPING SET
005026  013440  010000  005070          NOP                             %JNZRO  U.BST1  ; [QDA]IF SET THEN ALREADY MAPPED
005027  010547  007001  135573          ADD     #BUF.ST,R7\N,BAR        %CALL   S.LDRO  ;[QDA]GET STATUS WORD           ;[E121]
005030  033540  003010  135273          BIS     #BMAPDN,R0,BUF          %CALL   U.GETP  ;[QDA]SET MAPPING FLAG/GET ADDRESS OF FIRST SEG
005031  010545  007107  135573          ADD     #MAP.OF,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET OFFSET
005032  033141  000000  000000          BIS     R0,R1                                   ;[QDA]SET IN OFFSET
005033  010547  007007  135620          ADD     #BUF.UA,R7\N,BAR        %CALL   S.STR1  ;[QDA]STORE LOW ORDER BUS ADDRESS
005034  010547  007010  135613          ADD     #BUF.US,R7\N,BAR        %CALL   S.STQQ  ;[BDA]STORE IN HI ADDR

                                ; ***   THIS SECTION CALCULATES THE 22-BIT ADDRESS FOR THE NEXT MAPPING REGISTER.      ***
                                ; ***   IF THE INTERNAL CACHE IS EMPTY THE NEXT GROUP OF MAPPING REGISTERS IS READ     ***
                                ; ***   FROM HOST MEMORY.                                                             ***

005035  010545  007105  135614          ADD     #MAP.RD,UBAR\N,BAR      %CALL  S.DEC8   ;[QDA]DECREMENT NUMBER OF UNUSED READ PTE'S;[E121]
005036  013440  010000  015047          NOP                             %JNZRO U.MAP2   ;[QDA]IF SOME THEN O.K.         ;[E121]
005037  010545  007106  135574          ADD     #MAP.UR,UBAR\N,BAR      %CALL  S.LDR1   ;[QDA]SEE IF ANY LEFT TO GET    ;[E121]
005040  013740  017246  155051          MOV\F   #MAPFLG,BAR             %JZRO  U.MAP1   ;[QDA]IF NONE, CONTINUE - U.MAP1 WILL;[E121]
                                                                                        ;       FIND THE TRANSFER IS SINGLE-SEGMENT;[E121]
```



```
005041  013740  003001  010000          MOV     #1,BUF                                  ;[QDA]SET RETURN FLAG FOR U.PTEL;[E121]
005042  010545  007111  125573          ADD     #MAP.VO,UBAR\N,BAR      %CALL  S.LDRO   ;[QDA]GET LOW MAP REGISTER ADDRESS
005043  010545  007103  125616          ADD     #MAP.MO,UBAR\N,BAR      %CALL  S.STRO   ;[QDA]STORE LOW MAP REG ADDRESS
005044  010545  007112  125573          ADD     #MAP.V1,UBAR\N,BAR      %CALL  S.LDRO   ;[QDA]GET HIGH MAP REGISTER ADDRESS
005045  010545  007104  135616          ADD     #MAP.M1,UBAR\N,BAR      %CALL  S.STRO   ;[QDA]STORE HIGH MAP REGISTER ADDRESS
005046  013440  000000  115314          NOP                             %JMP   U.PTEL   ;[QDA]SET FLAG/IF SOME LEFT THEN GET WHAT WE CAN
005047  010545  007102  125574  U.MAP2: ADD     #MAP.NX,UBAR\N,BAR      %CALL  S.LDR1   ;[QDA]ELSE GET PTE POINTER
005050  030541  000002  135620          ADD     #PTELEN,R1              %CALL  S.STR1   ;[QDA]INCREMENT AND STORE BACK
                                                                                        ;[E121]
                                ;       Return here from U.PTEL or fall through from U.MAP2. MAP.NX(UBAR) -> next PTE in cache;[E121]
                                                                                        ;[E121]
005051  010545  007110  125574  U.MAP1: ADD     #MAP.S1,UBAR\N,BAR      %CALL  S.LDR1   ; get segment 1 size            ;[E121]
005052  010547  007012  125573          ADD     #BUF.BC,R7\N,BAR        %CALL  S.LDRO   ; get byte count                ;[E121]
005053  112140  000001  010000          CMP     R1,R0                                   ; see if transfer is single-segment;[E121]
005054  034466  020006  145065          CLR\T   IUAR                    %JCRY  10$      ; br if it is and clear seg2 wdct temp;[E121]
005055  013440  003001  010000          MOV     R1,BUF                                  ; reduce byte ct to seg size in buf ctl blk;[E121]
005056  131140  000001  000000          SUB     R1,R0                                   ; compute number of bytes in seg2;[E121]
005057  070140  000000  000000          ADD\L   R0,R0                                   ; shift left 7 to obtain        ;[E121]
005060  070140  000000  000000          ADD\L   R0,R0                                   ; word count in hi byte of IUAR ;[E121]
005061  070140  000000  000000          ADD\L   R0,R0                                   ;                               ;[E121]
005062  073446  000000  125273          MOV\L   R0,IUAR                 %CALL  U.GETP   ; get address of seg2 in R1 and Q;[E121]
005063  010547  007013  135620          ADD     #BUF.U1,R7\N,BAR        %CALL  S.STR1   ; store low order bus address   ;[E121]
005064  010547  007014  125613          ADD     #BUF.U2,R7\N,BAR        %CALL  S.STQQ   ; store high order bus address  ;[E121]
005065  010547  007011  135572  10$:    ADD     #BUF.GP,R7\N,BAR        %CALL  S.LDQQ   ; get group number from buf ctl blk;[E121]
005066  105640  000377  010000          BIC     #HIBYT,Q                                ; clear old seg2 length, if any ;[E121]
005067  033046  003006  000000          BIS     Q,IUAR,BUF                              ; substitute new seg2 length and store;[E121]

                                ;
                                U.BST1:
005070  010547  007012  125573          ADD     #BUF.BC,R7\N,BAR %CALL  S.LDRO  ; R0=BYTE COUNT FOR THIS SECTOR
005071  010547  007007  125600          ADD     #BUF.UA,R7\N,BAR %CALL  S.LDIU  ; IUAR=CURRENT LO HOST MEMORY ADDR
005072  010547  007010  125536          ADD     #BUF.US,R7\N,BAR %CALL  S.SXB1  ; R1=CURRENT HI HOST MEMORY ADDR
005073  010547  007002  125577          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDUB  ; [16K]UBAR=DATA BUFFER POINTER
005074  013740  007144  105575          MOV     #OPCODE,BAR      %JMP   S.LDR2  ; [16K]R2=OP CODE/RETURN


                                ; *** On unmapped transfers, make sure the mapping information is ***
                                ; *** cleared to prevent reads of the second sector from previous ***
                                ; *** mapped transaction.                                         ***

005075  010547  007014  010000  U.CLRM: ADD     #BUF.U2,R7\N,BAR                        ; clrs hi adr                   ;[E121]
005076  013740  003000  000000  U.CLM1: MOV     #0,BUF                                                                  ;[E121]
005077  010547  007011  125573          ADD     #BUF.GP,R7\N,BAR        %CALL  S.LDRO   ;[mjt05] get 2nd sec byte cnt
005100  034540  003377  115070          AND     #LOBYT,R0,BUF           %JMP   U.BST1   ;[mjt05] 2nd sec byte cnt

                                .PAGE
```

```
                                        .SBTTL  I/O PARAMETER CALCULATION ROUTINE
                                    ;;+
                                    ; ROUTINE NAME:
                                    ;       U.CPRM (CALCULATE I/O PARAMETERS)
                                    ;
                                    ; FUNCTIONAL DESCRIPTION:
                                    ;       THIS ROUTINE WILL CALCULATE THE PARAMETERS NECCESSARY FOR A
                                    ; SECTOR BUFFER WHEN IT IS FIRST ALLOCATED, BEFORE IT IS FILLED ON
                                    ; A WRITE OPERATION, OR AFTER IT IS EMPTIED ON A READ OPERATION.
                                    ; THE TRANSFER PARAMETERS CACULATED ARE AS FOLLOWS:
                                    ;       1. HOST TRANSFER ADDRESS
                                    ;       2. EXPECTED HEADER
                                    ;       3. CURRENT BUFFER ADDRESS
                                    ;       4. CURRENT GROUP ADDRESS AND IF A GROUP SWITCH IS NEEDED
                                    ;       5. CURRENT SECTOR ADDRESS
                                    ;       6. CURRENT HEAD ADDRESS
                                    ;
                                    ; INPUTS:
                                    ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                    ;       R7              POINTER TO SECTOR BUFFER CONTROL AREA
                                    ;       OPCODE          OPCODE OF CURRENT COMMAND - P.OPCD(SDI.PQ(UBAR))        ;[E121]
                                    ;
                                    ; OUTPUTS:
                                    ;       PARAMETERS CALCULATED/UPDATED IN SECTOR BUFFER AND SDI CONTROL BLOCK
                                    ;       *** NOTE - MUST PRESERVE R0,R1,R2 ***
                                    ;
                                    ;;-

                                    ; ***   SET DATA BUFFER POINTER FOR THIS SECTOR ***

005101  010545  007022  135602  U.CPRM: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; R10=MSCP PKT PTR

                                    ; ***   UPDATE BYTE TRANSFER COUNT REMAINING AND SET BUFFER BYTE COUNT ***       ;[E121]
                                                                                                                    ;[E121]
005102  010547  007012  010000          ADD     #BUF.BC,R7\N,BAR                ; POINT TO BUFFER BC WORD       ;[E121]
005103  103740  003002  010000          MOV     #<SECSZ*2),Q,BUF                ; ASSUME BUFFER IS FULL         ;[E121]
005104  010545  007015  125576          ADD     #SDI.XL,UBAR\N,BAR %CALL S.LDR3 ; R3 = LO XFER BYTE CT          ;[E121]
005105  132043  000003  000000          RSUB    Q,R3                            ; R3 = R3 - 512                 ;[E121]
005106  013440  023003  105115          MOV     R3,BUF          %JCRY   20$     ; UPDATE, BR IF R3 >= 512       ;[E121]
005107  010545  007016  135600          ADD     #SDI.XH,UBAR\N,BAR %CALL S.LDR6 ; R3 < 512 - R6 = HI XFER BYTE CT ;[E121]
005110  000043  010003  005114          ADD     R3,Q            %JNZRO  10$     ; Q = OLD LO BC, BR IF HI BC = 0 ;[E121]
005111  010547  007012  125613          ADD     #BUF.BC,R7\N,BAR %CALL  S.STQQ  ; BUFFER BC = OLD LO BC (PARTIAL BUF) ;[E121]
005112  010545  007015  010000          ADD     #SDI.XL,UBAR\N,BAR              ; SET LO BYTE CT = 0 - THIS IS  ;[E121]
005113  013740  003000  115115          MOV     #0,BUF          %JMP    20$     ; THE LAST BUFFER OF THE XFER   ;[E121]
005114  031446  000006  125624  10$:    DEC     R6              %CALL   S.STR6  ; BORROW FROM HI BYTE CT & UPDATE ;[E121]

                                    ; ***   UPDATE EXPECTED HEADER AND BYTES TO TRANSFER FOR THIS SECTOR ***

005115  010550  007016  135576  20$:    ADD     #S.LBNL,R10\N,BAR %CALL S.LDR3  ; R3=LO EXPECTED HEADER         ;[E121]
005116  010550  007017  135600          ADD     #S.LBNH,R10\N,BAR %CALL S.LDR6  ; R6=HI EXPECTED HEADER         ;[E121]
005117  010547  007003  010000          ADD     #BUF.HL,R7\N,BAR                                                ;[E121]
005120  120343  003003  000000          INCB    R3\0,R3,BUF                     ; SET LO EXPECTED HEADER AND INC ;[E121]
005121  010547  027004  105123          ADD     #BUF.HH,R7\N,BAR %JCRY  30$     ; PREPARE TO STORE HI HDR, CHECK CARRY ;[E121]
005122  013440  003006  115125          MOV     R6,BUF          %JMP    40$     ; SET HI EXPECTED HEADER, GO UPD LO ;[E121]
```



```
005123  120346  003006  000000  30$:    INCB    R6\0,R6,BUF                     ; SET HI EXPECTED FEADER AND INC ;[E121]
005124  010550  007017  135624          ADD     #S.LBNH,R10\N,BAR %CALL S.STR6  ; SAVE NEW HI HEADER            ;[E121]
005125  010550  007016  125622  40$:    ADD     #S.LBNL,R10\N,BAR %CALL S.STR3  ; SAVE NEW LO HEADER            ;[E121]
                                                                                                                ;[E121]
                                    ; ***   CHECK FOR GROUP SWITCH NEEDED ***                                    ;[E121]
                                                                                                                ;[E121]
005126  010550  007026  135576          ADD     #S.GRUP,R10\N,BAR %CALL S.LDR3  ; R3=NEW GROUP                  ;[E121]
005127  010547  007011  135572          ADD     #BUF.GP,R7\N,BAR %CALL  S.LDQQ  ; [BDA] S2 WORD COUNT IS FETCHED ;[E121]
005130  005640  000377  000000          BIC     #LOBYT,Q                        ; [BDA] STRIP OFF OLD GROUP     ;[E121]
005131  003043  003003  000000          BIS     R3,Q,BUF                        ; [BDA] AND STORE               ;[E121]
005132  010545  007005  125600          ADD     #SDI.UG,UBAR\N,BAR %CALL S.LDR6 ; R6=U.PROC GROUP WORD          ;[E121]
005133  026146  003003  000000          XOR     R3\0,R6,BUF                     ; RESET CURRENT GROUP, COMPARE GROUPS ;[E121]
005134  033766  010001  045135          MOV\T   #BGRUP,R6       %TNZRO          ; SET GROUP SWITCH FLG FOR STORING IN BUF.ST;[E121]

                                    ; ***   SET BUFFER TRANSFER ADDRESS & UPDATE HOST TRANSFER ADDRESS ***

005135  010550  007023  125572          ADD     #S.ADRH,R10\N,BAR %CALL S.LDQQ  ; Q=HOST HI TRANSFER ADDR       ;[E121]
005136  010547  007010  000000          ADD     #BUF.US,R7\N,BAR                                                ;[E121]
005137  105640  003300  010000          BIC     #BANOT,Q,BUF                    ; [BDA] SAVE HI ADDR BITS       ;[E121]
005140  010547  007001  125624          ADD     #BUF.ST,R7\N,BAR %CALL  S.STR6  ; NOW THAT GROUP IS SET, SET GROUP FLAG ;[E121]
005141  010550  007022  125576          ADD     #S.ADRL,R10\N,BAR %CALL S.LDR3  ; R3=HOST LO TRANSFER ADDR      ;[E121]
005142  133746  000002  010000          MOV     #<SECSZ*2>,R6                   ; Q=SECTOR SIZE IN BYTES        ;[E121]
005143  010547  007007  000000          ADD     #BUF.UA,R7\N,BAR                ; SET UP ST STORE LO ADR IN BUF CBLK ;[E121]
005144  020146  003003  000000          ADD     R3\0,R6,BUF                     ; STORE LOW ADR, ADD 512 TO IT  ;[E121]
005145  100240  020000  055147          INC\F   Q               %JNCRY  50$     ; IF CARRY THEN ADD TO HI ADDR  ;[E121]
005146  010550  007023  135613          ADD     #S.ADRH,R10\N,BAR %CALL S.STQQ  ; RESET HI TRANSFER ADDR        ;[E121]
005147  010550  007022  135624  50$:    ADD     #S.ADRL,R10\N,BAR %CALL S.STR6  ; RESET LO TRANSFER ADDR        ;[E121]

                                    ; ***   SET TRACK INTO BUFFER ***

005150  010550  007027  135572          ADD     #S.TRAK,R10\N,BAR %CALL S.LDQQ  ; Q=CURRENT TRACK               ;[E121]
005151  010547  007005  125613          ADD     #BUF.TA,R7\N,BAR %CALL  S.STQQ  ; STORE INTO BUFFER

                                    ; ***   CHECK FOR ACCESS/ERASE/REPLACE COMMANDS ***

005152  013740  007144  125600          MOV     #OPCODE,BAR     %CALL   S.LDR6  ; R6=OP CODE
005153                              ASSUME  <OP.ACC&OP.ERS&OP.RPL>&20,NE,0  ; ACCESS, ERASE, REPLACE HAVE BIT 4 SET ;[E121]
005153                              ASSUME  <OP.RD!OP.WR!OP.CMP!OP.ONL>&20,EQ,0 ;OTHERS DO NOT              ;[E121]
005153  014546  000020  010000          BIT     #20,R6                          ; [16K]IF NOT ACCESS/ERASE/REPLACE COMMAND;[E121]
005154  010547  017012  115157          ADD     #BUF.BC,R7\N,BAR %JZRO  U.CPME  ; [16K]  THEN CONTINUE          ;[E121]
005155  013740  003000  000000          MOV     #0,BUF                          ;   ELSE ZAP BYTE COUNT         ;[E121]
005156  010550  007023  125630          ADD     #S.ADRH,R10\N,BAR %CALL S.CLRB  ; AND HI UNIBUS ADDRESS

                                    ; ***   UPDATE SECTOR ADDRESS ***

005157                              U.CPME: ASSUME  LBNMSK,EQ,LOBYT             ; MAKE SURE MASK IS LO BYTE
005157  010545  007074  125606          ADD     #SDI.12,UBAR\N,BAR %CALL S.LLB6 ;[V05] R6=LBN'S/TRK 512 FORMAT
005160  010550  007030  135572          ADD     #S.SECS,R10\N,BAR %CALL S.LDQQ  ; Q=CURRENT SECTOR
005161  100240  000000  000000          INC     Q                               ; INCREMENT SECTOR ADDR
005162  012046  000006  000000          RSUBC   Q,R6\N                          ; IF NEW SECTOR ADDR LT MAX
005163  013260  023000  177777          MOV\T   Q,BUF           %RCRY           ;   THEN RETURN
005164  014240  003000  010000          CLR     Q,BUF                           ;   ELSE CURRENT SECTOR=0

                                    ; ***   UPDATE TRACK ADDRESS ***
```

```
005165  010545  007066  125606          ADD     #SDI.TG,UBAR\N,BAR %CALL S.LLB6 ;[V05] R6=TRKS/GROUP
005166  010550  007027  135572          ADD     #S.TRAK,R10\N,BAR %CALL S.LDQ0  ; Q=CURRENT TRACK
005167  100240  000000  135613          INC     Q                 %CALL  S.STQ0 ; UPDATE TRACK ADDRESS
005170  036046  000006  135607          XOR     Q,R6              %CALL  S.MLB6 ; IF CUR.TRK NEQ MAX
005171  010550  017027  037777          ADD     #S.TRAK,R10\N,BAR %RNZRO        ;   THEN RETURN
005172  104640  003377  000000          AND     #HIBYT,Q,BUF                    ; CURRENT TRACK=0

                                ; ***   UPDATE GROUP ADDRESS ***

005173  010550  007026  125600          ADD     #S.GRUP,R10\N,BAR %CALL S.LDR6  ; R6=CURRENT GROUP
005174  130446  000006  115624          INC     R6                %JMP   S.STR6 ; INCR BY 1,RESET/RETURN
                                        .PAGE
```



```
                                        .SBTTL  EDC CALCULATION ROUTINE
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.CKED (CHECK IF EDC FULLY CALCULATED)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE CHECKS IF THE EDC HAS BEEN FULLY CALCULATED BY THE
                                ; UNB.WR OR UNB.RD ROUTINES. IF THERE WAS A SHORT TRANSFER THEN THE EDC
                                ; CALCULATION IS COMPLETED. THIS ROUTINE WILL ALSO CALCULATE THE EDC
                                ; ON A FULL SECTOR BUFFER, THEN UBAR MUST POINT TO THE START OF THE BUFFER
                                ; AND R1 MUST EQUAL 69.
                                ;
                                ; INPUTS:
                                ;       R2      POINTER TO SECTOR BUFFER WHERE EDC CALCULATION STOPPED
                                ;               (START OF BUFFER IF WHOLE BUFFER EDC DESIRED)
                                ;       R3      EDC CALCULATED TO DATE (69. IF WHOLE BUFFER EDC DESIRED)
                                ;       R7      POINTER TO SECTOR BUFFER CONTROL AREA
                                ;
                                ; OUTPUTS:
                                ;       R3      EDC FOR THIS SECTOR
                                ;       R2,BAR  POINT TO THE EDC WORD IN THE SECTOR BUFFER
                                ;       UBAR    RESTORED TO POINT AT SDI CONTROL BLOCK
                                ;
                                ;;-
005175  010547  007006  135577  U.CKED: ADD     #BUF.SD,R7\N,BAR %CALL  S.LDUB  ; RESTORE UBAR TO SDI CTRL BLK PTR
005176  010547  007012  135574          ADD     #BUF.BC,R7\N,BAR %CALL  S.LDR1  ; R1=BYTE COUNT
005177  116541  000002  000000          XOR     #<SECSZ*2>,R1\N                 ; QUICK CHECK FOR FULL SECT             ;[E121]
005200  013460  017002  177777          MOV\T   R2,BAR          %RZRO           ; IF SO, SET BAR AND GET OUT            ;[E121]
005201  010547  007011  125573          ADD     #BUF.GP,R7\N,BAR %CALL  S.LDR0  ;[BDA]GET GROUP FOR SEG 2               ;[E121]
005202  055540  000377  125223          BIC\R   #LOBYT,R0       %CALL   U.R1R0  ; [BDA] CLEAR ANY GROUP BITS            ;[E121]
005203  053440  010000  105206          MOV\R   R0,R0           %JZRO   10$                                             ;[E121]
005204  053440  000000  135223          MOV\R   R0,R0           %CALL   U.R1R0                                          ;[E121]
005205  053440  000000  135223          MOV\R   R0,R0           %CALL   U.R1R0                                          ;[E121]
                                10$:                                                                                    ;[E121]
005206  030141  000000  000000          ADD     R0,R1                           ;[QDA]ADD BYTE COUNT                    ;[E121]
005207  173740  000001  000000          MOV\L   #SECSZ,R0                       ; R0=SECTOR SIZE IN BYTES
005210  151140  000001  000000          SUB\R   R1,R0                           ; R0=EDC LOOPS NEEDED
005211  013440  017002  137777          MOV     R2,BAR          %RZRO           ; [16K]IF EQ 0 THEN RETURN
005212  010545  007025  134121          ADD     #SDI.0M,UBAR\N,BAR %CALL CKWROM ; [U52EC1]IF NOT WRITE/ERASE/REPLACE
005213  013440  017002  105220          MOV     R2,BAR          %JZRO   U.CKEB  ;   THEN DO READ/ACCESS/COMPARE EDC     ;[E121]

                                ; ***   FINISH WRITE/ERASE EDC CALCULATION ***

005214  073443  010003  167777  U.CKEA: SHF\LF  R3              %RZRO           ; [16K]IF ZERO THEN RETURN/ELSE XOR 0
005215  013740  003000  000000          MOV     #0,BUF                          ; [16K]ZAP BUFFER
005216  130442  007002  010000          INC     R2,R2,BAR                       ; [16K] UPDATE BUFFER POINTER AND STORE 0
005217  031440  000000  105214          DEC     R0              %JMP    U.CKEA  ; DECR LOOP CNT/LOOP

                                ; ***   FINISH READ/ACCESS/COMPARE EDC CALCULATION ***
                                ; ***   ENTRY POINT FOR COMPARE EDC CALCULATION, R3=SEED,R2 & BAR=BUFFER PTR, R0=WORD COUNT ***

005220  076503  010003  167777  U.CKEB: XOR\LF  (BUF),R3        %RZRO           ; [16K]IF ZERO THEN RETURN/ELSE DO XOR
005221  130442  007002  010000          INC     R2,R2,BAR                       ; UPDATE BUFFER POINTER
```

```
005222  031440  000000  115220            DEC     R0              %JMP    U.CKEB  ; DECR LOOP CNT/LOOP                    ;[E121]
005223  053440  000000  137777  U.R1R0:   MOV\R    R0,R0           %RET                                                    ;[E121]
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.CKTC (CHECK TRANSFER COUNT)
                                ;
                                ; FUNCTIONAL DESCIPTION:
                                ;       THIS ROUTINE WILL LOAD THE CURRENT HOST TRANSFER COUNT (HI AND LO)
                                ; AND OR THEM TOGETHER AS A TEST FOR TRANSFER COMPLETE.
                                ;
                                ; INPUTS:
                                ;       UBAR    POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;       TRANSFER COUNT LO IN R1
                                ;       TRANSFER COUNT HI IN R0
                                ;       ZERO CONDITION CODE SET ON OR OF R0,R1
                                ;
                                ;;-
005224  010545  007016  125573  U.CKTC: ADD     #SDI.XH,UBAR\N,BAR %CALL S.LDR0 ; R0=HI TRANSFER COUNT REMAINING        ;[E121]
005225  010545  017015  115574          ADD     #SDI.XL,UBAR\N,BAR %JZR0 S.LDR1 ; R1=LO TRANSFER COUNT REMAINING        ;[E121]
005226  033701  000003  010000          MOV     (BUF),R1                        ; R1=LO TRANSFER COUNT REMAINING        ;[E121]
005227  013141  000000  137777          OR      R0,R1\N         %RET            ; [U52EC1]OR TOGETHER IN R1/RETURN      ;[E121]
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.XCMP (TRANSFER COMPLETE ROUTINE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PROCESSES THE COMPLETION OF A MSCP COMMAND.
                                ;
                                ; INPUTS:
                                ;       UBAR            POINTER TO SDI CONTROL BLOCK
                                ;
                                ; OUTPUTS:
                                ;
                                ;;-

005230  010545  007022  135602  U.XCMP: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; R10=PTR TO ACTIVE PKT
005231  013440  000000  112166          NOP                       %JMP    U.OPCD  ; [16K]GO TO OP CODE DISPATCHER

                                        .PAGE
```



```
                                ;
                                ;       UBAR -> SDI CONTROL BLOCK
                                ;       R10  =  MSCP PACKET POINTER
                                ;
005232                          U.MINT: ASSUME  PAGESZ,EQ,512.                          ;[QDA]
005232  033740  000001  000000          MOV     #1,R0
005233  010545  007115  135616          ADD     #MAP.ST,UBAR\N,BAR      %CALL S.STR0    ; NOTE THAT MAPPING INIT

005234  010545  007103  135630          ADD     #MAP.M0,UBAR\N,BAR      %CALL   S.CLRB  ;[QDA]CLEAR MAP REG 0
005235  010545  007104  125630          ADD     #MAP.M1,UBAR\N,BAR      %CALL   S.CLRB  ;[QDA]CLEAR MAP REG 1
005236  010545  007105  135630          ADD     #MAP.RD,UBAR\N,BAR      %CALL   S.CLRB  ;[QDA]CLEAR #READ PTE'S
005237  010550  007012  135574          ADD     #P.BUF0,R10\N,BAR       %CALL   S.LDR1  ;[QDA]WORD 0 BUF DESCRIPT
005240  010545  007107  000000          ADD     #MAP.OF,UBAR\N,BAR                      ;[QDA]POINT TO OFFSET FIELD
005241  115541  003376  000000          AND     #OFFMSK,R1\N,BUF                        ;[QDA]GET OFFSET
005242  010545  007111  125620          ADD     #MAP.V0,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]VIRT ADD WORD 0
005243  010550  007013  125574          ADD     #P.BUF1,R10\N,BAR       %CALL   S.LDR1  ;[QDA]WORD 1 BUF DESCIPT
005244  010545  007112  125620          ADD     #MAP.V1,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]VIRT ADD WORD 1
005245  010550  007014  135574          ADD     #P.BUF2,R10\N,BAR       %CALL   S.LDR1  ;[QDA]WORD 2 BUF DESCIPT
005246  010545  007113  135620          ADD     #MAP.V2,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]MAP REG BASE WORD 1
005247  010550  007015  125574          ADD     #P.BUF3,R10\N,BAR       %CALL   S.LDR1  ;[QDA]WORD 3 BUF DESCIPT
005250  010545  007114  125620          ADD     #MAP.V3,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]MAP REG BASE WORD 2
                                        ; debug temp kjk
005251  010550  007002  125574          add     #p.crf0,r10\n,bar       %call   s.ldr1  ;display ref number lo
005252  010550  007003  135574          add     #p.crf1,r10\n,bar       %call   s.ldr1  ;display ref number hi
                                        ; end debug
005253  010550  007010  125574          ADD     #P.BCN0,R10\N,BAR       %CALL   S.LDR1  ;[QDA]FOR DIVIDE
005254  010550  007011  125573          ADD     #P.BCN1,R10\N,BAR       %CALL   S.LDR0  ;[QDA] DITTO                                    ;[E121]
005255                                  ASSUME PAGESZ,EQ,512.
005255  133742  000002  124127          MOV     #PAGESZ,R2                      %CALL   DIV512  ;[QDA]GET NUMBER OF PTE'S NEEDED;[E121]
005256  010545  007106  125620          ADD     #MAP.UR,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]STORE NUMBER NEEDED
005257  010545  007107  125574          ADD     #MAP.OF,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET OFFSET
005260  030140  000001  000000          ADD     R1,R0                                   ;[QDA]GET (OFFSET+REM)
005261  103740  010002  105265          MOV     #PAGESZ,Q               %JZR0   U.MIN1  ;[QDA]GET PAGE SIZE/IF ZERO THEN DONE
005262  131040  000000  000000          SUB     Q,R0                                    ;[QDA]COMPARE WITH PAGE SIZE
005263  033760  020001  155265          MOV\T   #1,R0                   %JCRY   U.MIN1  ;[QDA]LESS THAN OR EQUAL TO PAGE SIZE ADD 1
005264  033740  000002  000000          MOV     #2,R0                                   ;[QDA]GREATER THAN PAGE SIZE ADD 2
005265  010545  007106  135574  U.MIN1: ADD     #MAP.UR,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET UNREAD PTE'S
005266  030141  000000  135620          ADD     R0,R1                   %CALL   S.STR1  ;[QDA]STORE IT BACK/RETURN
005267  010545  007107  135573          ADD     #MAP.OF,UBAR\N,BAR      %CALL   S.LDR0  ;[QDA]GET OFFSET
005270  010545  007110  000000          ADD     #MAP.S1,UBAR\N,BAR                      ;[QDA]POINT TO LOC FOR SEGMENT 1 SIZE
005271  131040  000000  135616          SUB     Q,R0                    %CALL   S.STR0  ;[QDA]STORE SEGMENT 1 SIZE
005272  013440  000000  115314          NOP                             %JMP    U.PTEL  ;[QDA]READ IN FIRST SET OF PTE'S

                                        .PAGE
```

```
                                ;
                                ;
                                ;       THE FOLLOWING CODE PERFORMS A LEFT SHIFT OF A 32 BIT QUANTITY.
                                ;       IT DEPENDS ON THE FACT THAT THE Q REGISTER SHIFTS IN ZEROES AND
                                ;       SHIFTS OUT TO THE BIT BUCKET AND THAT THE REGISTER BANK DOES TRUE ROTATES.
                                ;
005273                          ASSUME  MAPVAL,EQ,BIT15
                        U.GETP:
005273  010545  007102  135573          ADD     #MAP.NX,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET POINTER TO CURRENT ENTRY
005274  013440  007000  135574          MOV     RO,BAR                  %CALL   S.LDR1  ;[QDA]GET LOW ORDER PTE AT THAT LOCATION
005275  110440  007000  135573          INC     RO,BAR                  %CALL   S.LDRO  ;[QDA]GET HIGH ORDER PTE AT THAT LOC;[E121]
005276  034540  030177  005311          AND     #BIT07-1,RO             %JNMSB  U.INVM  ;ISOLATE HIGH PFN IN RO,        ;[E122]
                                                                                        ;IF NO VALID BIT THEN ERROR     ;[E122]
005277  070140  000000  135310          ADD\L   RO,RO                   %CALL   U.R1R1  ;Shift RO left 9, R1 right 7    ;[E121]
005300  070140  000000  135310          ADD\L   RO,RO                   %CALL   U.R1R1                                  ;[E121]
005301  070140  000000  135310          ADD\L   RO,RO                   %CALL   U.R1R1                                  ;[E121]
005302  070140  000000  135310          ADD\L   RO,RO                   %CALL   U.R1R1                                  ;[E121]
005303  073440  000000  135310          MOV\L   RO,RO                   %CALL   U.R1R1  ; RO<15:9> has PTE<22:16>       ;[E121]
005304  053441  000001  125310          MOV\R   R1,R1                   %CALL   U.R1R1                                  ;[E121]
005305  105541  000376  010000          BIC     #177000,R1,Q                            ; Q<8:0> = PTE<15:7>            ;[E121]
005306  134541  000376  000000          AND     #177000,R1                              ; R1<15:9> = PTE<6:0>           ;[E121]
005307  003040  000000  127777          BIS     RO,Q                    %RET            ; Q = PTE<22:7>                 ;[E121]
                                                                                                                        ;[E121]
005310  053441  000001  137777  U.R1R1: MOV\R   R1,R1                   %RET                                            ;[E121]

                                ; At this point, the valid bit in the mapping register has
                                ; been detected to be off.  This is a Host Buffer Access Error
                                ; (see MSCP sec. 8.4) Since it has not yet accessed the bus, it
                                ; does not return a log error message.  This error is treated like
                                ; the other host buffer errors that do not involve bus accesses i.e.
                                ; odd byte count (sc.odb) and odd transfer address (sc.odt).

                                ; The jump into the par table is strictly to unwind the stack.

                                ; begin [qda]
000025                          INVPOP  = POP2 - 7400
005311  013740  002025  127777  u.invm: mov     #invpop,par             %ret    ; drop u.getp ret on floor
005312  033743  000251  010000  u.inva: mov     #<st.hst+sc.imr>,r3             ; load error code [rae9-c119]
005313  034442  000602  127777          clr     r2      @rupf           %ret    ; return to whoever called
                                                                                ; u.bset.  from there jump
                                                                                ; to error handling.
                                ; end   [qda]


                                .PAGE
```



```
005314  010545  007103  135574  U.PTEL: ADD     #MAP.MO,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET ADDRE OF NEXT UNREAD PTE
005315  010545  017104  015341          ADD     #MAP.M1,UBAR\N,BAR      %JNZRO  U.PTL2  ;[QDA]IF NOT ZERO NOT THE FIRST TIME
005316  033701  000003  010000          MOV     (BUF),R1                                ;[QDA]GET HIGH ADDR OF NEXT UNREAD PTE
005317  013440  010000  015341          NOP                             %JNZRO  U.PTL2  ;[QDA]IF NOT ZERO NOT THE FIRST TIME
005320  010545  007111  135572          ADD     #MAP.VO,UBAR\N,BAR      %CALL   S.LDQQ  ;[QDA]GET LOW ORDER MAP REG NUM
005321  010545  007112  125601          ADD     #MAP.V1,UBAR\N,BAR      %CALL   S.LDR7  ;[QDA]GET HIGH ORDER MAP REF NUM
005322  033740  000007  000000          MOV     #7.,RO                                  ;[QDA]SHIFT COUNT
005323  145547  010200  145326  U.PTL3: BIC\RQF  #BIT15,R7              %JZRO   U.PTL4  ;[QDA]DOUBLE RIGHT SHIFT R7,Q/CLEAR BIT15 R7
005324  103660  040200  045325          BIS\T   #BIT15,Q                %TLSB           ;[QDA]IF LSB OF R7 THE SET MSB OF Q
005325  031440  000000  105323          DEC     RO                      %JMP    U.PTL3  ;[QDA]DECR LOOP COUNT/JUMP
005326  005640  000003  000000  U.PTL4: BIC     #MAPMSK,Q                               ;[QDA]CLEAR BITS 0 AND 1
005327  010545  007113  135573          ADD     #MAP.V2,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET LOW ORDER BASE ADDR
005330  010545  007114  135574          ADD     #MAP.V3,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET HIGH ORDER BASE ADDR
005331  030040  000000  000000          ADD     Q,RO                                    ;[QDA]ADD LOW ORDERS
005332  130461  020001  145333          INC\T   R1                      %TCRY           ;[QDA]TAKE CARE OF CARRY
005333  033446  000000  010000          MOV     RO,IUAR                                 ;[BDA]SET LOW ORDER
005334  010545  007111  125616          ADD     #MAP.VO,UBAR\N,BAR      %CALL   S.STRO  ;[QDA]STORE MAP REGISTER ADDRESS LOW
005335  030141  000007  010000          ADD     R7,R1                                   ;[QDA]GET HIGH ORDER
005336  003441  000001  135541          MOV     R1,Q                    %CALL   S.SXAB  ;[QDA]SAVE R1 AND SET HIGH BITS
005337  010545  007112  125613          ADD     #MAP.V1,UBAR\N,BAR      %CALL   S.STQQ  ;[QDA]STORE MAP REGISTER ADDRESS HIGH
005340  013440  000000  105345          NOP                             %JMP    U.PTL5
005341  010545  007103  125573  U.PTL2: ADD     #MAP.MO,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET LOW MAPPING REGISTER
005342  033446  000000  010000          MOV     RO,IUAR                                 ;[BDA]SET UP ADDR REGS
005343  010545  007104  125574          ADD     #MAP.M1,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET HIGH MAPPING REGISTER
005344  003441  000001  135541          MOV     R1,Q                    %CALL   S.SXAB  ;[QDA]SAVE R1/SET IN HIGH BITS
005345  010545  007106  125573  U.PTL5: ADD     #MAP.UR,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET UNREAD PTE'S
005346  112540  000042  010000          CMP     #MEMSZ,RO                               ;[QDA]COMPARE WITH AVAILABLE MEM
005347  033760  020104  055351          MOV\T   #MEMSZ*2,RO             %JNCRY  10$     ;[QDA][cho3]READ WHAT WE CAN OR
005350  175540  000200  010000          BIC\L   #BIT15,RO                               ;[QDA]READ ALL THAT'S LEFT
005351  013740  007244  135616  10$:    MOV     #MAPSAV,BAR             %CALL   S.STRO  ;[QDA]SAVE RO FOR LATER USE
005352  104640  000300  000000          AND     #BANOT,Q                                ; [BDA] MAKE SURE BITS 14,15 OF HIGH ARE ZERO
005353  033447  000005  000000          MOV     UBAR,R7                                 ;[QDA]SAVE UBAR
005354  010545  007101  125577          ADD     #MAP.CH,UBAR\N,BAR      %CALL   S.LDUB  ;[QDA]POINT TO BUFFER
005355  013440  000000  125635          NOP                             %CALL   UNB.RD  ;[QDA]READ IN THE PTE'S
005356  033767  010026  052071          MOV\T   #ER.MRR,R7              %JNZRO  U.UERR  ;[QDA]ERROR IF NOT ZERO [rae4-c119]
005357  033445  000007  000000          MOV     R7,UBAR                                 ;[QDA]RESTORE UBAR
005360  013740  007244  135575          MOV     #MAPSAV,BAR             %CALL   S.LDR2  ;[QDA]RESTORE NUMBER OF PTE'S READ
005361  010545  007106  125573          ADD     #MAP.UR,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET NUMBER OF UNREAD PTE'S
005362  055542  000001  000000          BIC\R   #BIT00,R2                               ;[QDA]DIV R2/2 TO GET NUMBER OF PTE'S READ/UNUSED
005363  131140  000002  135616          SUB     R2,RO                   %CALL   S.STRO  ;[QDA]SUBTRACT NUM READ/UNUSED FROM TOT AND STORE
005364  010545  007105  135621          ADD     #MAP.RD,UBAR\N,BAR      %CALL   S.STR2  ;[QDA]RESET NUMBER OF PTE'S READ AND UNUSED
005365  010545  007111  125573          ADD     #MAP.VO,UBAR\N,BAR      %CALL   S.LDRO  ;[QDA]GET LOW BASE OF MAP REG
005366  010545  007112  135574          ADD     #MAP.V1,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET HIGH BASE OF MAP REG
005367  070142  000002  000000          ADD\L   R2,R2                                   ;[QDA]ADJUST FOR QUAD WORD
005370  030140  000002  000000          ADD     R2,RO                                   ;[QDA]BUMP IT
005371  130461  020001  145372          INC\T   R1                      %TCRY           ;[QDA]ADD THE CARRY
005372  010545  007111  125616          ADD     #MAP.VO,UBAR\N,BAR      %CALL   S.STRO  ;[QDA]STORE IT BACK
005373  010545  007112  125620          ADD     #MAP.V1,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]STORE IT BACK
005374  010545  007101  125574          ADD     #MAP.CH,UBAR\N,BAR      %CALL   S.LDR1  ;[QDA]GET ADDRESS OF BEGINNING OF CACHE
005375  010545  007102  135620          ADD     #MAP.NX,UBAR\N,BAR      %CALL   S.STR1  ;[QDA]MAKE THE NEXT ENTRY BE AT THE BEGINNING/RETURN
005376  013740  007246  135574          MOV     #MAPFLG,BAR             %CALL   S.LDR1  ;[QDA]POINT TO MAPPING FLAG
005377  014440  013000  127777          CLR     RO,BUF                  %RZRO           ;[QDA]MAKE SURE ZERO RETURN CODE/DO RETURN IF CALLED
005400  010545  007006  000000          ADD     #SDI.UB,UBAR\N,BAR                      ;[QDA]RESTORE R7
005401  033707  000003  115051          MOV     (BUF),R7                %JMP    U.MAP1  ;[QDA]AND JUMP BACK TO MAPPING ROUTINE
```

```
                                        .PAGE
```



```
                                        .SBTTL  U.PROC XFC ROUTINES
                                    ;;+     19-JUNE-82      UDA52 MICROCODE
                                    ; ROUTINE NAME:
                                    ;       U.XFCP
                                    ;
                                    ; FUNCTIONAL DESCRIPTION:
                                    ;       THIS ROUTINE IS ENTERED WHEN THE U PROCESSOR IDLE LOOP
                                    ; DETECTS A D.PROC REQUEST FOR XFC SERVICE. IT ANALYZES THE XFC
                                    ; CODE IN R11 AND VECTORS TO THE APPROPRIATE XFC HANDLER. WHEN THE ROUTINE
                                    ; IS COMPLETE IT WILL CLEAR DXFC IN RLL THEREBY RELEASING THE D.PROC.
                                    ;
                                    ; INPUTS:
                                    ;       R11             XFC CODE FROM D.PROC
                                    ;       RLL             SYSTEM STATUS REGISTER
                                    ;
                                    ; OUTPUTS:
                                    ;       R11             XFC STATUS AS APPROPRIATE
                                    ;       RLL             CLEARED DXFC BIT
                                    ;
                                    ;;-

005402  033740  010324  127777  U.XFCP: MOV     #<UP.XFC-XFCUPR-1>&LOBYT,R0 %RZR0 ; IF NO XFC THEN RETURN
                                    ; ***   R11 MUST BE PRESERVED BECAUSE THIS CODE MAY BE ENTERED MANY TIMES ***
005403  011551  000022  000000          SUBC    #XFCMAX,R11\N                   ; IF XFC EXCEEDS MAX LIMIT
005404  033771  030021  045405          MOV\T   #XFCFIN,R11     %TNMSB          ;   THEN XFC=XFC DONE VALUE
005405  030140  000011  010000          ADD     R11,R0                          ; R0=JUMP TABLE ADDRESS
005406  013440  002000  010000          MOV     R0,PAR                          ; GO DO U.PROC XFC ROUTINE
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC11
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE READ AND WRITE UNIBUS MEMORY XFC'S.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           LO ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 1           HI ORDER UNIBUS ADDRESS
                                ;       DM REGISTER 2           NUMBER OF WORDS TO TRANSFER
                                ;       DM REGISTER 3           DM BUFFER ADDRESS
                                ;
                                ; OUTPUTS:
                                ;       DATA READ/WRITTEN FROM/TO THE HOST
                                ;       DM REGISTER 0           EDC CALCULATED ON DATA READ/WRITE
                                ;       DM REGISTER 1           0 - SUCCESSFUL
                                ;                               1 - NONEXISTENT UNIBUS MEMORY
                                ;                               2 - UNIBUS PARITY ERROR
                                ;
                                ;;-

                                UXFC11::MOV     #1,R11          %JMP    UXFCRW  ; EXECUTED IN JUMP TABLE
                                UXFC12::CLR     R11             %JMP    UXFCRW  ; EXECUTED IN JUMP TABLE
005407  013740  007007  125600  UXFCRW: MOV     #DMREG0,BAR     %CALL   S.LDIU  ; IUAR=LO UNIBUS ADDRESS
005410  013740  007001  125536          MOV     #DMREG1,BAR     %CALL   S.SXB1  ; R1=HI UNIBUS ADDRESS
005411  013740  007003  135577          MOV     #DMREG3,BAR     %CALL   S.LDUB  ; UBAR=BUFFER ADDRESS
                                ; ***   VALIDATE DM BUFFER ADDRESS - MUST BE >= DM.BEG ***
005412  033740  000365  010000          MOV     #DM.BEG+DMSTR&LOBYT,R0          ; [16K]CONSTRUCT DM.BEG IN R0
005413  133540  000002  000000          BIS     #DM.BEG&HIBYT,R0                ; [16K]R0=DM LOWER LIMIT
005414  121140  007005  010000          SUB     UBAR\0,R0,BAR                   ; IF UBAR =< R0 (DM.BEG)
005415  033767  020014  152071          MOV\T   #ER.DMX,R7      %JCRY   U.UERR  ;   THEN FATAL ERROR
005416  013740  007002  135573          MOV     #DMREG2,BAR     %CALL   S.LDR0  ;   ELSE R0=WORD COUNT
005417  013451  000011  135634          TST     R11             %CALL   UXFUIO  ; GO DO READ/WRITE
005420  033271  010011  055421          MOV\T   Q,R11           %TNZR0          ; IF ERROR THEN SET R11
005421  013740  007007  135620          MOV     #DMREG0,BAR     %CALL   S.STR1  ; DM REG 0 GETS CALCULATED EDC
005422  135544  000020  137777          BIC     #DXFC,RLL       %RET            ; RELEASE D.PROC/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC13
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE PERFORM ECC CORRECTION XFC.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO BUFFER CONTROL AREA
                                ;       DM REGISTER 2           SIZE OF SECTOR IN WORDS
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 0           NUMBER OF CORRECTIONS MADE
                                ;       DM REGISTER 1           0 - SUCCESSFUL COMPLETION
                                ;                               1 - UNSUCCESSFUL COMPLETION
                                ;
                                ;;-

                                UXFC13::MOV     #DMREG0,BAR     %JMP    UXFC13  ; EXECUTED IN JUMP TABLE
005423  033707  000603  126450          MOV     (BUF),R7 @RUPF  %CALL   ECCC    ; GO DO ECC CORRECTION
005424  013740  007134  125573          MOV     #ECOUNT,BAR     %CALL   S.LDR0  ; R0=#SYMBOLS IN ERROR
005425  013740  007007  135616          MOV     #DMREG0,BAR     %CALL   S.STR0  ; SAVE IN DM REG 1
005426  033451  000003  115453          MOV     R3,R11          %JMP    UXF16B  ; R11=ERROR STATUS/EXIT
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC14
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE SEND DATA XFC.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO DATA BUFFER TO SEND
                                ;       DM REGISTER 1           SIZE OF DATA BUFFER IN WORDS
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 1           0 - SUCCESSFUL COMPLETION
                                ;                               1 - I/O ERROR ON DATA TRANSFER
                                ;       DM REGISTER 2           NUMBER OF WORDS TRANSFERRED
                                ;
                                ;;-

                                UXFC14:;MOV     #MRQUE,R7,BAR                           ; EXECUTED IN JUMP TABLE
005427  033710  000003  010000  UXF14A: MOV     (BUF),R10                               ; R10=I/O QUEUE HEAD
005430  013740  017001  137777          MOV     #DMREG1,BAR     %RZRO                   ; IF NO I/O CMD THEN RETURN
005431  073701  000003  000000          MOV\L   (BUF),R1                                ; R1=XFC REQUEST BYTE COUNT
005432  010550  007010  135575          ADD     #P.BCNO,R10\N,BAR %CALL S.LDR2          ; R2=I/O PKT BYTE COUNT
005433  111142  000001  000000          SUB     R1,R2\N                                 ; IF XFC REQ LEQ THEN RESET
005434  033462  023001  145435          MOV\T   R1,R2,BUF       %TCRY                   ; [16K] THEN P.BCNO=XFC BYTE COUNT
005435  013740  007002  125621          MOV     #DMREG2,BAR     %CALL   S.STR2          ; [16K]DM R2=BYTE COUNT
005436  013740  007007  125577          MOV     #DMREGO,BAR     %CALL   S.LDUB          ; UBAR=DM BUFFER ADDRESS
005437  004240  000000  133375          CLR     Q               %CALL   U.MI0B          ; [16K]GO CALC UNIBUS I/O PARAMS
005440  016567  010146  075634          XOR\T   #MRQUE,R7\N     %CNZRO  UXFUIO          ; IF RO NEQ O THEN GO DO I/O
005441  033760  010011  045443          MOV\T   #ST.HST,RO      %JNZRO  UXF14B          ; IF ERROR THEN CONTINUE
005442  033740  000000  010000          MOV     #ST.SUC,RO                              ;   ELSE RO=SUCCESS CODE
005443  033441  000010  125453  UXF14B: MOV     R10,R1          %CALL   UXF16B          ; SAVE R10 IN R1/CLR 'DXFC'
005444  013440  007007  123713          MOV     R7,BAR          %CALL   U.ULNK          ; GO UNLINK PKT
005445  033450  000001  112420          MOV     R1,R10          %JMP    U.C2EP          ; RESTORE PKT PTR/GO CHG TO END PKT/RETURN
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC15
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE RECEIVE DATA XFC.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO DATA BUFFER TO FILL
                                ;       DM REGISTER 1           SIZE OF DATA BUFFER IN WORDS
                                ;
                                ; OUTPUTS:
                                ;       DM REGISTER 1           0 - SUCCESSFUL COMPLETION
                                ;       DM REGISTER 2           NUMBER OF WORDS TRANSFERRED
                                ;
                                ;;-

                                UXFC15:;MOV     #MWQUE,R7,BAR   %JMP    UXF14A  ; EXECUTED IN JUMP TABLE
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC16
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE CONVERT LBN, RBN, DBN, XBN
                                ; CONVERSION XFC.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0           POINTER TO 12 WORD CONVERSION TABLE
                                ;       DM REGISTER 1           POINTER TO SUBUNIT CHARACTERISTICS BLOCK
                                ;
                                ; OUTPUTS:
                                ;       CONVERSION DATA IN TABLE POINTED TO BY DM REGISTER 0
                                ;       R11             ZERO FOR ALL CONVERSIONS ARE SUCCESSFUL BY DEFAULT
                                ;
                                ;;-

                                UXFC16:;MOV     #5,R3           %JMP    UXFC16  ; EXECUTED IN JUMP TABLE
005446  013740  007007  135601         MOV     #DMREG0,BAR     %CALL   S.LDR7  ; R7=PTR TO CONVERSION TABLE
005447  033746  000040  126672         MOV     #CNVTV1,R6      %CALL   ECMOVE  ; R6=PTR TO U.CNVT PARAMS/GO SET THEM UP
005450  013740  007001  135576         MOV     #DMREG1,BAR     %CALL   S.LDR3  ; R3=SUBUNIT CHAR PTR
005451  131543  000063  125612         SUB     #SDI.H1,R3      %CALL   S.CL11  ; ADJUST START OF SUBUNIT CHAR/CLR R11
005452  033445  000007  122430         MOV     R7,UBAR         %CALL   U.CNVT  ; UBAR=RESULT PTR/GO DO CONVERSION
005453  135544  000020  137777 UXF16B: BIC     #DXFC,RLL       %RET            ; RELEASE D.PROC/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC17
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE EXIT DM MODE XFC.
                                ;
                                ; INPUTS:
                                ;       DM REGISTER 0   - 0             NORMAL DM EXIT
                                ;                       - NONZERO       FATAL DM ERROR
                                ;
                                ; OUTPUTS:
                                ;       EXIT FROM DM MODE
                                ;
                                ;;-

                                UXFC17:;MOV     #<16384.-DM.BEG>&LOBYT,R0 %JMP UXFC17; [16K]EXECUTED IN JUMP TABLE
005454  013740  007007  125574         MOV     #DMREG0,BAR     %CALL   S.LDR1  ; R0=DM EXIT FLAG
005455  033767  010014  042071         MOV\T   #ER.DMX,R7      %JNZRO  U.UERR  ; IF NONZERO THEN FATAL AND STOP
005456  133540  000075  135534         BIS     #<16384.-DM.BEG>&HIBYT,R0 %CALL S.FUNB; [16K]R0=UPPER HALF OF LOOP COUNT
005457  033441  000005  125544         MOV     UBAR,R1         %CALL   S.ZBUF  ; [16K]GO ZAP THE BUFFER
005460  013440  000000  131072         NOP                     %CALL   S.SSDI  ; GO RESET ALL SDI CONTROL BLKS
                                ; ***   NOTE THE FOLLOWING INSTRUCTION DOES A JUMP TO U.STRT INSTEAD OF
                                ; ***   RETURNING AS IT SHOULD. THE 2901 STACK DOES RAP AROUND SO THIS
                                ; ***   MISUSE SHOULD NOT CAUSE ANY HARM
005461  135544  000120  101500         BIC     #<DMODE!DXFC>,RLL %JMP  U.STRT  ; [16K]EXIT DMODE&XFC/RESTART
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       UXFC18
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL PERFORM THE INCREMENT STUD STATUS XFC.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       INCREMENTED STUD STATUS
                                ;
                                ;;-

                                UXFC18:;MOV     #DM.BEG+DSTSL&LOBYT,R0  %JMP UXFC18 ; [16K]EXECUTED IN VECTOR TABLE
005462  133540  007002  010000          BIS     #DM.BEG+DSTSL&HIBYT,R0,BAR      ; [16K]R0,BAR=PTR TO LO DUST STATUS
005463  033706  000003  125611  5$:     MOV     (BUF),R6        %CALL   S.INC6  ; INCREMENT LO/HI STUD STATUS           [E122]
005464  013440  023006  005466          MOV     R6,BUF          %JNCRY  10$     ; IF NO CARRY THEN DONE                 [E122]
005465                                  ASSUME  DSTSH,EQ,DSTSL+1                ; [16]MAKE SURE STATUS SEQUENTIAL
005465  110440  007000  105463          INC     R0,BAR          %JMP    5$      ; [16K]  ELSE INCR HI WORD
                                10$:                                            ;                                       [E122]
                                                                                ; generate new report of SW version, HW ver [E122]
                                                                                ; and controller model                  [E122]
                                                                                ; DMREG1 (alias error status) =         [E122]
                                                                                ;   <15:10>     = controller model number[E122]
                                                                                ;   <9:6>       = sw code version       [E122]
                                                                                ;   <5:0>       = hw rev level          [E122]
005466  013740  007275  010000          MOV     #BUFP49,BAR                     ;get hardware rev level                 [E122]
005467  033711  000003  000000          MOV     (BUF),R11                       ;                                       [E122]
005470  033551  000300  000000          BIS     #<CODVER*BIT06>&LOBYT,R11               ;or in low byte bits of SW ver  [E122]
                                                                                ;or in model bits and hi bits of SW ver
005471  133551  000114  105453          BIS     #<<CMODEL*BIT10>!<CODVER*BIT06>>&HIBYT,R11 %JMP UXF16B
                                .PAGE
```



```
                                ; *-------------------------------------------------------------------------*
                                ; *      RRRRRR     00000   U      U  TTTTTTT     III     N      N  EEEEEEE   *
                                ; *      R     R   0     0  U      U     T         I      NN     N  E         *
                                ; *      R     R   0     0  U      U     T         I      N N    N  E         *
                                ; *      RRRRRR    0     0  U      U     T         I      N  N   N  EEEEE     *
                                ; *      R   R     0     0  U      U     T         I      N   N  N  E         *
                                ; *      R    R    0     0  U      U     T         I      N    NN  E         *
                                ; *      R     R    00000    UUUUU      T        III     N     N  EEEEEEE   *
                                ; *-------------------------------------------------------------------------*
                                        .SBTTL  U.PROC AND D.PROC SUBROUTINES
                                ;;+     29-JULY-83      UDA50-A VERSION 5 MICROCODE
                                ; ROUTINE NAMES:
                                ;       P.LOCK
                                ;       UNLOCK
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       P.LOCK WILL WAIT FOR THE ALTERNATE PROCESSOR TO SET THE 'PLOCK' BIT
                                ; IN THE CONTROL REGISTER IMAGE (RLL). IT WILL THEN CLEAR THE 'PLOCK' BIT
                                ; IN THE RLL TO LOCK OUT THE ALTERNATE PROCESSOR.
                                ;       UNLOCK WILL SET THE 'PLOCK' BIT IN RLL, EFFECTIVELY UNLOCKING THE
                                ; ALTERNATE PROCESSOR.
                                ;
                                ; INPUTS:
                                ;       RLL             CONTROL REGISTER IMAGE
                                ;
                                ; OUTPUTS:
                                ;       P.LOCK - ALTERNATE PROCESSOR LOCKED OUT OF MUTEXED DATA AREAS
                                ;       UNLOCK - ALTERNATE PROCESSOR FREE TO USE MUTEXED DATA AREAS
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-
005472                                  ASSUME  PLOCK,EQ,BIT15                  ; PLOCK MUST BE MSB
005472  013440  020000  137777  P.LCKA: NOP                     %RCRY           ; IF OTHER PROC FREE THEN RETURN
005473  050144  000004  115472  P.LOCK: ADD\R   RLL,RLL          %JMP    P.LCKA  ; IF OTHER PROCESSOR MUTEXING

005474  133544  000200  137777  UNLOCK: BIS     #PLOCK,RLL      %RET            ; SET PLOCK/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.GMST (U.PROC GET MUTEXED SDI STATUS)
                                ;       D.GMST (D.PROC GET MUTEXED SDI STATUS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL LOCK OUT THE OPPOSITE PROCESSOR (USING THE
                                ; PLOCK BIT IN RLL), LOAD THE SDI STATUS (FROM SDI.ST) INTO R0 OR R11,
                                ; AND RETURN. IT SHOULD BE NOTED THAT IT IS THE CALLER'S RESPONSIBILITY
                                ; TO FREE THE OPPOSITE PROCESSOR
                                ; BY EITHER CALLING UNLOCK OR BY PERFROMING THE FOLLOWING INSTRUCTION:
                                ;       BIS     #PLOCK,RLL
                                ;
                                ; INPUTS:
                                ;       UBAR POINTING TO SDI CONTROL BLOCK OF INTEREST
                                ;       DBAR POINTING TO SDI CONTROL BLOCK OF INTEREST
                                ;
                                ; OUTPUTS:
                                ;       SDI STATUS IN R0
                                ;       SDI STATUS IN R11
                                ;       LOCKED OUT U.PROC OR D.PROC
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005475  010545  027000  115573          ADD     #SDI.ST,UBAR\N,BAR %JCRY S.LDR0 ; THEN R0 = SDI STATUS / RETURN
005476  050144  000004  105475  U.GMST: ADD\R   RLL,RLL          %JMP    .-1     ; IF PLOCK (BIT15) IS ONE
```



```
                                ;;+                                                                     [rae04]
                                ; ROUTINE NAME:                                                         [rae04]
                                ;       U.MODL                                                          [rae04]
                                ;                                                                       [rae04]
                                ; FUNCTIONAL DESCRIPTION                                                [rae04]
                                ;       This routine ORs in the Controller Model number into the Controller ID  [rae04]
                                ; and stores it.  This is jumped to from U.SCID which is located in a fixed     [rae04]
                                ; location to allow manufacturing to blast the controller serial number into    [rae04]
                                ; the PROM.  When the QDA got assigned model "13" the code required changing    [rae04]
                                ; to accomodate the low order bit.                                      [rae04]
                                ;;-                                                                     [rae04]
                                U.MODL:                                 ;                               [rae04]
005477  033540  000022  103370          BIS     #CMODEL,R0      %JMP    STROLG  ; Set the model/Store/return    [rae04]
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.RRR1
                                ;       S.RRR7
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES ROTATE THE VALUE IN REGISTER R1/R7 RIGHT THE
                                ; NUMBER OF TIMES INDICATED IN REGISTER Q. THIS IS THE UNIBUS PROCESSOR'S
                                ; ROTATE RIGHT ROUTINE.
                                ;
                                ; INPUTS:
                                ;       R1/R7           VALUE TO BE ROTATED RIGHT
                                ;       Q               THE NUMBER OF TIMES TO ROTATE R1 RIGHT
                                ;
                                ; OUTPUTS:
                                ;       R1/R7           INPUT VALUE ROTATED RIGHT
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005500  053441  010001  127777          SHF\R   R1              %RZRO           ; SHIFT R1 RIGHT/IF Q = 0 THEN RETURN
005501  001240  000000  115500  S.RRR1: DEC     Q               %JMP    .-1     ; DECR Q

005502  053447  010007  127777          SHF\R   R7              %RZRO           ; SHIFT R7 RIGHT/IF Q = 0 THEN RETURN
005503  001240  000000  105502  S.RRR7: DEC     Q               %JMP    .-1     ; DECR Q

005504  003740  000010  105501  S.SWB1: MOV     #8.,Q           %JMP    S.RRR1  ; BYTE SWAP ROUTINE
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.RELC (RELEASE BUFFER CONTROL BLOCKS)
                                ;       S.RELF (RELEASE DATA BUFFER)                                    ;[E121]
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       S.RELC - THIS ROUTINE WILL RELEASE THE RING OF BUFFER CONTROL
                                ; BLOCKS ASSIGNED TO A SDI CONTROL BLOCK USING THE RING POINTER STORED
                                ; IN SDI.UB.
                                ;       S.RELF - THIS ROUTINE WILL RELEASE THE DATA BUFFER POINTED TO    ;[E121]
                                ; BY (BUF), IF THE CONTENTS OF (BUF) ARE ZERO THEN NO DATA BUFFER EXISTS. ;[E121]
                                ;                                                                       ;[E121]
                                ; INPUTS:                                                               ;[E121]
                                ;       UBAR    POINTER TO SDI CONTROL BLOCK (S.RELC AND S.RELF)        ;[E121]
                                ;       BAR     POINTS TO POINTER TO BUFFER (S.RELF)                    ;[E121]
                                ;
                                ; OUTPUTS:
                                ;       UPDATED BUFFER AVAILABLE COUNTER
                                ;       R2 AND Q ARE USED AS TEMPORARY REGISTERS
                                ;
                                ; STACK LEVEL:
                                ;       ONE USED
                                ;
                                ;;-

005505  010545  007006  125572  S.RELC: ADD     #SDI.UB,UBAR\N,BAR %CALL S.LDQQ ; Q=PTR TO BUFFER CHAIN
005506                          ASSUME  BUF.NL,EQ,0                             ; MAKE SURE BUF.NL IS ZERO
005506  013740  013000  137777  S.RLBA: MOV     #0,BUF          %RZRO           ; [16K]IF ZERO THEN RETURN/CLR PTR
005507  010640  007002  125575          ADD     #BUF.BP,Q\N,BAR %CALL   S.LDR2  ; [16K]R2=ADD OF DATA BUFR
005510  013760  013000  075516          MOV\T   #0,BUF          %CNZRO  S.RELD  ; [16K]CLEAR BUF.BP/RELEASE BUFFER
005511  013240  007000  010000          MOV     Q,BAR                           ; BAR=BUFFER POINTER
005512  003700  010003  015506          MOV     (BUF),Q         %JNZRO  S.RLBA  ; [16K]IF PTR NEQ 0 THEN LOOP
005513  013440  000000  127777          NOP                     %RET            ; [16K]  ELSE RETURN

                                ; ***   RELEASE BUFFER POINTED TO BY (BUF) ***                          ;[E121]
005514  033702  000003  010000  S.RELF: MOV     (BUF),R2                        ; [V05][16K]R2=BUFFER POINTER
005515  013740  013000  137777          MOV     #0,BUF          %RZRO           ; [16K]INDICATE BUFFER RELEASED
005516  025544  000001  010000  S.RELD: BIC     #1,RLL\0                        ; [16K]IF BUFFER LOCK SET
005517  010544  047003  155516          ADD\F   #3,RLL\N,BAR    %JNLSB  .-1     ; [16K]  THEN WAIT
005520  013440  003002  010000          MOV     R2,BUF                          ; [16K]  ELSE PUT PTR BACK ON STACK
005521  030544  000003  137777          ADD     #3,RLL          %RET            ; [16K]UPDATE STK PTR/REMOVE BLOCK/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.SDIO  (GETS ADDRESS OF SDI.1 IN R0)
                                ;       S.SDI1  (GETS ADDRESS OF SDI.1 IN UBAR)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL LOAD THE ADDRESS OF SDI.1 IN EITHER UBAR
                                ; OR R0.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       UBAR            ADDRESS OF SDI.1
                                ;       R0              ADDRESS OF SDI.1
                                ;
                                ;;-

005522  033745  000355  010000  S.SDI1: MOV     #SDI.1&LOBYT,UBAR               ; UBAR=LO SDI.1 ADDRESS
005523  133545  000002  127777          BIS     #SDI.1&HIBYT,UBAR %RET          ; SET IN HI ADDRESS/RETURN

005524  033740  000355  010000  S.SDIO: MOV     #SDI.1&LOBYT,R0                 ; R0=LO SDI.1 ADDRESS
005525  133540  000002  127777          BIS     #SDI.1&HIBYT,R0 %RET            ; SET IN HI ADDRESS/RETURN

005526  033751  000355  010000  S.SD11: MOV     #SDI.1&LOBYT,R11                ; R11=LO SDI.1 ADDRESS
005527  133551  000002  127777          BIS     #SDI.1&HIBYT,R11 %RET           ; SET IN HI ADDRESS/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.FMB1
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE FORMS THE ADDRESS OF THE FIRST SECTOR DATA BUFFER.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       R0              ADDRESS OF FIRST DATA BUFFER
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005530  033740  000255  000000  S.FMB1: MOV     #BUFR1.&LOBYT,R0                ; R0=BUFR #1 ADDR LO
005531  133540  000012  137777          BIS     #BUFR1.&HIBYT,R0 %RET           ; R0 =BUFFER ADDR/RETURN
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.FBC1
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE FORMS THE ADDRESS OF THE FIRST BUFFER CONTROL BLOCK.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       R0              ADDRESS OF FIRST BUFFER CONTROL BLOCK
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005532  033740  000055  010000  S.FBC1: MOV     #BUFBEG&LOBYT,R0            ; [16K]R0=BUFR CNTRL BLK #1 ADDR LO
005533  133540  000004  127777          BIS     #BUFBEG&HIBYT,R0 %RET       ; [16K]R0=BUFR CNTRL BLK ADDR/RETURN
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.FUNB
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE FORMS THE ADDRESS OF DM.BEG IN UBAR
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       UBAR            ADDRESS OF DM.BEG
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005534  033745  000352  000000  S.FUNB: MOV     #DM.BEG&LOBYT,UBAR          ; [16K]UBAR=LO ADDR OF DM.BEG
005535  133545  000002  127777          BIS     #DM.BEG&HIBYT,UBAR %RET     ; [16K]UBAR=HI ADDR OF DM.BEG
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.SXAB (SET EXTENDED ADDRESS BITS)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       ROUTINE S.SXAB WILL SET THE EXTENDED ADDRESS BITS FROM R1
                                ; INTO THE CONTROL REGISTER.
                                ;
                                ; INPUTS:
                                ;       R1              EXTENDED UNIBUS ADDRESS BITS
                                ;
                                ; OUTPUTS:
                                ;       UPDATED CR
                                ;
                                ; STACK LEVEL:
                                ;       NONE USED
                                ;
                                ;;-

005536  033446  000006  010000  S.SXB1: MOV     IUAR,IUAR                       ; SHOW IUAR
005537  033701  000003  105541          MOV     (BUF),R1        %JMP    S.SXAB  ; GET R1
005540  031461  020001  055541  S.AXAB: DEC\T   R1              %TNCRY          ; IF NO CARRY THEN DECR R1
140000                          BANOT := ^C<BA>
005541  135541  000300  000000  S.SXAB: BIC     #<BANOT>,R1                     ; ISOLATE HI ADDRESS BITS
005542  013740  007262  115620          MOV     #SAVUAR,BAR     %JMP    S.STR1  ; [mjto5] SAVE R1 IN SAVUAR
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.ZBUF (ZERO BUFFER)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       ROUTINE S.ZBUF WILL ZERO THE BUFFER AREA DEFINED BY R0 (LENGTH)
                                ; AND R1(BUFFER POINTER).
                                ;
                                ; INPUTS:
                                ;       R0              WORDS TO ZERO COUNT
                                ;       R1              BUFFER POINTER
                                ;
                                ; OUTPUTS:
                                ;       BUFFER ZEROED
                                ;
                                ; STACK LEVEL:
                                ;       ONE LEVEL USED
                                ;
                                ;;-

005543  013740  013000  137777  SZBUFA: MOV     #0,BUF          %RZRO           ; ZAP WD / IF R0=0 THEN RETURN          ;[E121]
005544  120341  007001  010000  S.ZBUF: INCB    R1\0,R1,BAR                     ; BUMP ADDRESS, LOAD BAR                ;[E121]
005545  031440  000000  105543          DEC     R0              %JMP    SZBUFA  ; DECREMENT WORD COUNT/LOOP
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       DIVD
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE DIVIDES A 31 BIT UNSIGNED DIVIDEND BY A 15 BIT
                                ; UNSIGNED DIVISOR AND RETURNS A QUOTIENT AND REMAINDER.
                                ;
                                ; INPUTS:
                                ;       R1 = DIVIDEND LOWER
                                ;       R0 = DIVIDEND UPPER
                                ;       R2 = DIVISOR
                                ;
                                ; OUTPUTS:
                                ;       R1 = QUOTIENT
                                ;       R0 = REMAINDER
                                ;       Q IS USED AS A TEMPORARY REGISTER
                                ;
                                ;;-

005546  034440  000000  000000  DIVDO:  CLR     R0                              ; ZAP R0
005547  003740  000021  115554  DIVD:   MOV     #17.,Q          %JMP    DIVB    ; Q = 17.
005550  130140  010000  145557  DIVX:   ADDC\F  R0,R0           %JZR0   DIVXIT  ; IF NEQ 0 THEN CONTINUE
005551  131140  000002  000000  DIVA:   SUB     R2,R0                           ; SUB DIVISOR FROM HI DIVIDEND
005552  030160  020002  045554          ADD\T   R2,R0           %JNCRY  DIVB    ; IF NO CARRY THEN LOOP
005553  130141  000001  115555          ADDC    R1,R1           %JMP    DIVC    ; LEFT SHIFT LO DIVIDEND
005554  030141  000001  010000  DIVB:   ADD     R1,R1                           ; DOUBLE R1
005555  001240  020000  105550  DIVC:   DEC     Q               %JCRY   DIVX    ; IF CARRY THEN CONTINUE
005556  030160  010000  055551          ADD\T   R0,R0           %JNZR0  DIVA    ; IF Q NEQ 0 THEN LOOP
005557  013440  000000  127777  DIVXIT: TST     R0              %RET            ; TEST R0/RETURN
                                .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       MULT
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE MULTIPLIES TWO 16 BIT NUMBERS TOGETHER AND RETURNS
                                ; A 16 BIT RESULT. THE MULTIPLICATION PERFORMED IS UNSIGNED.
                                ;
                                ; INPUTS:
                                ;       R6      MULTIPLICAND
                                ;       Q       MULTIPLIER
                                ;
                                ; OUTPUTS:
                                ;       R7 IS THE LO ORDER RESULT
                                ;
                                ;;-

005560  003700  000003  000000  MULBYT: MOV     (BUF),Q                         ; Q=BUFFER
005561  004640  000377  010000          AND     #LOBYT,Q                        ; ISOLATE LO BYTE
005562  034447  000007  105564  MULT:   CLR     R7              %JMP    MULTB   ; ZAP LO ORDER RESULT
005563  053446  010006  127777  MULTA:  MOV\R   R6,R6           %RZR0           ; IF EQ 0 THEN RETURN
005564  053446  000006  010000  MULTB:  MOV\R   R6,R6                           ; DIVIDE R6 BY 2
005565  030067  040007  055566          ADD\T   Q,R7            %TLSB           ; IF LSB THEN ADD Q TO R7
005566  165546  000200  115563          BIC\LQ  #BIT15,R6       %JMP    MULTA   ; ZAP HI ORDER BIT/CONTINUE
                                .PAGE
```

```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       S.LDCR,S.LDQQ,ETC.
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL LOAD THE REGISTER NUMBER SPECIFIED AS THE LAST
                                ; TWO CHARACTERS OF THE SUBROUTINE NAME, I.E. S.LD11 WILL LOAD R11 FROM BUF.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO THE BUFFER ADDRESS WHOSE DATA IS DESIRED
                                ;                       TO BE LOADED IN THE REGISTER
                                ;
                                ; OUTPUTS:
                                ;       REGISTER LOADED WITH CONTENTS OF BUFFER POINTED TO BY BAR
                                ;
                                ;;-
005567  033710  000003  010000  S.LDCR: MOV     (BUF),R10                       ; [NEW ECC]R10=(BUF)
005570  113550  007074  000000          BIS     #LGADR,R10\N,BAR                ; [NEW ECC](BAR)=(BUF)
005571  013450  000010  127777          TST     R10             %RET            ; [NEW ECC]SELECT LOG BUFFER AREA
005572  003700  000003  127777  S.LDQQ: MOV     (BUF),Q         %RET            ; Q=(BUF)
005573  033700  000003  127777  S.LDR0: MOV     (BUF),R0        %RET            ; R0=(BUF)
005574  033701  000003  137777  S.LDR1: MOV     (BUF),R1        %RET            ; R1=(BUF)
005575  033702  000003  137777  S.LDR2: MOV     (BUF),R2        %RET            ; R2=(BUF)
005576  033703  000003  127777  S.LDR3: MOV     (BUF),R3        %RET            ; R3=(BUF)
                                S.LDUB:
005577  033705  000003  127777  S.LDR5: MOV     (BUF),UBAR      %RET            ; R5/UBAR=(BUF)
                                S.LDIU:
005600  033706  000003  127777  S.LDR6: MOV     (BUF),IUAR      %RET            ; R6/IUAR=(BUF)
005601  033707  000003  137777  S.LDR7: MOV     (BUF),R7        %RET            ; R7=(BUF)
005602  033710  000003  137777  S.LD10: MOV     (BUF),R10       %RET            ; R10=(BUF)

                                ; ***   SPECIALTY ROUTINES ***

005603  014500  000003  137777  S.BIT0: BIT     (BUF),R0        %RET            ; BIT-TEST BUFFER WITH R0
005604  033700  000003  000000  S.LLB0: MOV     (BUF),R0                        ; R0=(BUF)
005605  034540  000377  137777  S.MLB0: AND     #LOBYT,R0       %RET            ; ISOLATE LO BYTE/RETURN
005606  033706  000003  000000  S.LLB6: MOV     (BUF),R6                        ; R6=(BUF)
005607  034546  000377  137777  S.MLB6: AND     #LOBYT,R6       %RET            ; ISOLATE LO BYTE/RETURN
005610  016500  000003  127777  S.XOR0: XOR     (BUF),R0\N      %RET            ; COMPARE BUFFER WITH R0
005611  130446  000006  127777  S.INC6: INC     R6              %RET            ; INCREMENT R6/RETURN
005612  034451  000011  127777  S.CL11: CLR     R11             %RET            ; ZAP R11/RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAMES:
                                ;       S.STQQ,S.STR0,ETC.
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE ROUTINES WILL STORE THE THE REGISTER NUMBER SPECIFIED AS THE
                                ; LAST TWO CHARACTERS OF THE SUBROUTINE NAME INTO THE BUFFER LOCATION POINTED
                                ; TO BY BAR, I.E. S.ST11 WILL STORE R11 INTO THE BUFFER.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO BUFFER AREA WHERE REGISTER DATA IS
                                ;                       DESIRED TO BE STORED
                                ;
                                ;;-
005613  013240  003000  127777  S.STQQ: MOV     Q,BUF           %RET            ; (BUF)=Q
005614  033700  000003  000000  S.DECB: MOV     (BUF),R0                        ; R0=(BUF)
005615  031440  010000  167777  S.DEC0: DEC\F   R0              %RZR0           ; IF EQ 0 THEN RETURN ELSE DECR R0
005616  013440  003000  127777  S.STR0: MOV     R0,BUF          %RET            ; (BUF)=R0
005617  034541  000377  000000  S.SLB1: AND     #LOBYT,R1                       ; ISOLATE LO BYTE OF R1
005620  013440  003001  137777  S.STR1: MOV     R1,BUF          %RET            ; (BUF)=R1
005621  013440  003002  137777  S.STR2: MOV     R2,BUF          %RET            ; (BUF)=R2
005622  013440  003003  127777  S.STR3: MOV     R3,BUF          %RET            ; (BUF)=R3
                                S.STUB:
005623  013440  003005  127777  S.STR5: MOV     R5,BUF          %RET            ; (BUF)=R5/UBAR
                                S.STIU:
005624  013440  003006  127777  S.STR6: MOV     IUAR,BUF        %RET            ; (BUF)=R6/IUAR
005625  013440  003007  137777  S.STR7: MOV     R7,BUF          %RET            ; (BUF)=R7
005626  013440  003010  137777  S.ST10: MOV     R10,BUF         %RET            ; (BUF)=R10
005627  013440  003011  127777  S.ST11: MOV     R11,BUF         %RET            ; STORE R11 INTO BUFFER
                                        .PAGE
```

```
                                ;;+
                                ; ROUTINE NAME:
                                ;       S.CLRB  (CLEAR BUFFER LOCATION)
                                ;       INIT1   (SET BUFFER LOCATION TO 1)
                                ;       INITM1  (SET BUFFER LOCATION TO -1)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THE S.CLRB ROUTINE WILL STORE ZERO INTO THE BUFFER POINTED TO
                                ; BY BAR.
                                ;       THE INIT1 ROUTINE PUTS A 1 IN THE MEMORY LOCATION POINTED TO BY BAR.
                                ;       THE INITM1 ROUTINE PUTS A -1 IN THE MEMORY LOCATION POINTED TO BY BAR.
                                ;       ***NOTE THESE ROUTINES CAN BE CALLED BY EITHER PROCESSOR BECAUSE THEY
                                ; USE NO REGISTERS.
                                ;
                                ; INPUTS:
                                ;       BAR             POINTER TO BUFFER LOCATION TO CLEAR/SET TO 1/-1
                                ;
                                ; OUTPUTS:
                                ;       CLEARED BUFFER LOCATION (S.CLRB)
                                ;       1 IN BUFFER LOCATION (INIT1)
                                ;       -1 IN BUFFER LOCATION (INITM1)
                                ;
                                ;;-

005630  014440  003000  137777  S.CLRB: CLR     ,\N,BUF         %RET            ; (BUF) = 0/RETURN

005631  013740  003001  137777  INIT1:  MOV     #1,BUF          %RET            ; INITIALIZE TO 1/RETURN

005632  017740  003000  137777  INITM1: COM     #0,BUF          %RET            ; INITIALIZE TO 177777 / RETURN
                                        .PAGE
```



```
                                ;;+
                                ; ROUTINE NAME:
                                ;       U.PROC PATCH AREA
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THESE NOP'S WILL PROVIDE A SPACE FOR U.PROC CHANGES SUCH THAT A SMALL
                                ; CHANGE COULD MINIMIZE THE AFFECTED PROMS.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ;;-
                                ;                                               ; [16K]REMOVE PATCH SPACE
                                ;       NOP                                     ; [ECO#1]U.PROC PATCH SPACE
                                ;       NOP                                     ; [ECO#1]U.PROC PATCH SPACE
                                ;       NOP                                     ; [ECO#1]U.PROC PATCH SPACE
                                ;       NOP                                     ; [ECO#1]U.PROC PATCH SPACE
                                ;                                               ; [ECO#2]ADJUST PATCH SPACE
                                        .PAGE
```

```
                        ; ***   ROUTINE UNB.RS RESETS THE UNIBUS,POLLING REGISTER,ETC ***
005633  013440  000660  110252  UNB.RS: NOP             @RBCAI  %JMP    TSTRTN  ; [BDA] RESET BCAI AND RETURN
                        .page
```



```
                        ;*************************************************************************
                        ;                                                                        *
                        ;  BI.RD        (BI read subroutine)                                     *
                        ;                                                                        *
                        ;*************************************************************************
                        ;                                                                        *
                        ;  This is the bus read routine design. (Note:  by convention, an        *
                        ;  unqualified read or write refers to a host read or write.  The        *
                        ;  bus read routine is called by the host write routine since the        *
                        ;  data flow is from the host to storage; from the controller's          *
                        ;  perspective, from host memory to controller buffers.)                 *
                        ;                                                                        *
                        ;  Assumptions and interface:                                            *
                        ;                                                                        *
                        ;  1.  Mapping has already been done.  Addresses are contiguous.         *
                        ;                                                                        *
                        ;  2.  Inputs (same as unb.rd):                                          *
                        ;              ubar     :  starting internal buffer ptr (dest)           *                ;[E121]
                        ;              iuar,uar:   low 16 bits of external bus address         *                ;[E121]
                        ;                          for start of transfer.  Identical copies.   *
                        ;              r0       :  byte count for the transfer                  *
                        ;                                                                        *
                        ;       BI Conventions                                                   *
                        ;                                                                        *
                        ;              bcai(d/s) : where addresses and data go into and out of   *
                        ;              breg      : sets up necessary BI logic signals            *
                        ;                                                                        *
                        ;   3. Outputs (same as unb.rd):                                        *
                        ;              q        :  error code - CC's reflect value of Q         *                ;[E121]
                        ;                          If r0 <> 0,                                 *
                        ;                          q = 1 means nxm error (time-out)            *
                        ;                          q = 2 means parity error                    *
                        ;              upf      :  Set upf implies timeout, not set             *                ;[E121]
                        ;                          implies parity.  See u.uner                 *                ;[E121]
                        ;              r0       :  number of words not transfered               *
                        ;                          On success, transfer will be 0.             *
                        ;              r1       :  EDC                                          *
                        ;                                                                        *
                        ;  4.  Side-effects:                                                    *
                        ;              iuar, uar are changed due to convention differences.     *
                        ;              ubar,bar  are advanced.  Ubar left at last word + 1      *
                        ;                              to maintain existing interface.           *
                        ;              Host data in buffer pointed to by ubar on entry.          *
                        ;              r2 and r3 are not preserved.                              *
                        ;              r7 and r10 used as scratch but original contents saved    *
                        ;                              and  restored                             *
                        ;                                                                        *
                        ;*************************************************************************
                        .page
```

```
;*****************************************************************************
;  Notes:                                                                     *
;                                                                             *
;  1.  Entry to this routine occurs at subroutine nest level 3.              *
;      Therefore,  only one more level of calling is allowed from this       *
;      code (2911 constraint).                                                *
;                                                                             *
;                                                                             *
;                                                                             *
;  Subroutines used:                                                          *
;                                                                             *
;       CHKBDR   :  Check for chunk boundary, i.e., where lower 16 bits       *
;                   overflow.                                                 *
;       BI.TO    :  BI transfer timeout routine                               *                ;[E121]
;                                                                             *
;  Entry points:                                                              *
;                                                                             *
;       a.  the usual entry point is QB.RD for Qbus, BI.RD for BI.            *                ;[E121]
;       b.  UNB.RD is retained to insure that                                 *                ;[E121]
;           code shared by the uda/qda/bda use                                *                ;[E121]
;           the correct bus read routine.                                     *
;       c.  UN.BRC is used to avoid resetting                                 *
;           the edc residue.                                                  *
;       d.  UXFUIO extended function code entry                               *
;                                                                             *
;*****************************************************************************
.page
```



```
005634  034451  010011  116242  UXFUIO: CLR     R11,R11                 %JZRO   BI.WR   ; XFC ENTRY
                                  UNB.RD:                                              ; UDA COMPATIBLE
                                  QB.RD:                                               ; QDA COMPATIBLE
005635  033741  000105  010000  BI.RD:  MOV     #EDSEED,R1                            ; R1 WILL CONTAIN EDC
005636  013740  007034  000000  UN.BRC: MOV     #FAILUR,BAR                           ; IN CASE BI STOP OCCURS
005637  013740  003013  010000          MOV     #ER.MST,BUF                           ; LOAD ERROR CODE INTO LAST FAILURE

005640  013740  007044  125625  RD.DIAG: MOV    #CNVTV3,BAR             %CALL   S.STR7  ; SAVE R7 AND R10
005641                          ASSUME  <CNVTV4&17>,EQ,<CNVTV3&17>+1
005641  033450  003010  010000          MOV     R10,R10,BUF
005642  013740  007262  125576          MOV     #SAVUAR,BAR             %CALL   S.LDR3  ; R3 = IUAR EXTENSION                   ;[E121]
                                                                                       ;[E121]
005643  013740  116221  056235  BI.RLP: MOV\F   #<B.LED+B.NABO+B.NRTY>,BREG %JSTOP BI.STO ; IF STOP, THEN HANG HERE      ;[E121]
                                                                                       ; ELSE SET EVEN ADDRESS, SELF-TEST;[E121]
005644  053442  000660  000000          MOV\R   R0,R2           @RBCAI                 ; OK, NO ABORT, NO RETRY          ;[E121]
005645  055542  010003  106112          BIC\R   #3,R2                   %JZRO   BIREOK  ; R2 = # LONGWORDS                ;[E121]
005646  155542  000140  000000          BIC\R   #60000,R2                               ; R2 = # QUADWORDS & clear some bits;[E121]
005647  014546  010016  116113          BIT     #16,IUAR                %JZRO   BIR.NO  ; R2 = # OCTAWORDS & clear rest of them;[E121]
005650  053450  010006  006114          MOV\R   IUAR,R10                %JNZRO  BIR.SO  ; BR IF NO OCTAWORDS              ;[E121]
005651  152447  000006  126170          NEG\R   IUAR,R7                 %CALL CHKBDR    ; BR IF NOT ON OCTAWORD BOUNDARY;[E121]
005652  114544  000100  010000          BIT     #DMODE,RLL                             ; OK TO READ OWDS - CHECK PG OVFLO;[E121]
005653  013440  010000  006142          NOP                             %JNZRO  DM.RD   ; IF WE ARE IN DIAG MODE          ;[E121]
005654  024447  005566  000000          CLR     IUAR\0,R7,BCAID @LADD                  ; USE SPECIAL ROUTINE             ;[E121]
005655  113543  005300  000000          BIS     #0W,R3\N,BCAID                         ; LOAD LO ADDR, SET R7=0          ;[E121]
005656  130447  000767  010000          INC     R7              @HADD                  ; SET OWORD XFER BITS IN HI ADR   ;[E121]
005657                          ASSUME  BIRDCM,EQ,1                                    ; STORE IN BCAI/SET R7 = 1        ;[E121]
                                                                                       ;[E121]
                                  ; TO BE HERE WE MUST HAVE SOME OCTAWORDS TO READ AND BE ON AN OCTAWORD BOUNDARY ;[E121]
                                                                                       ;[E121]
005657  021342  005547  010000          DECB    R7\0,R2,BCAID   @LCOM                  ; LOAD READ COMMAND, predecr ct   ;[E121]
005660  073447  010420  115666          MOV\L   R0,R7           @GDRD   %JZRO   30$    ; Start 1st owd rd, copy WC to BC temp;[E121]
                                                                                       ; one octawd is a special case    ;[E121]
005661  035547  000017  010000          BIC     #17,R7                                 ; assume no errors will occur,    ;[E121]
005662  030146  000007  000000          ADD     R7,IUAR                                ; update BI addr too              ;[E121]
005663  004545  020017  005565          AND     #17,UBAR,Q              %JNCRY  20$    ; Q = block offset of internal buf;[E121]
005664  130443  000003  000000          INC     R3                                     ; Handle 64K seg ovflo            ;[E121]
005665  000640  002004  000000  20$:    ADD     #BIRDBR-7400,Q,PAR                     ; Branch into ultra-optimized     ;[E121]
                                                                                       ; read loop                       ;[E121]
005666  035547  000017  010000  30$:    BIC     #17,R7                                 ; Semi-duplicate of above code    ;[E121]
005667  030146  000007  000000          ADD     R7,IUAR                                ; update BI addr to end of loop   ;[E121]
005670  131545  020010  016061          SUB     #8.,UBAR                %JNCRY  BIR.LW  ; bias UBAR for loop              ;[E121]
005671  130443  000003  106061          INC     R3                      %JMP    BIR.LW                                    ;[E121]
                                  .page                                                                                  ;[E121]
```

```
                                                                                                                    ;[E121]
                                ;       Ultra-optimized read loop takes maximum advantage of the auto-increment     ;[E121]
                                ;       feature of storing into local buffer memory. There are actually eight       ;[E121]
                                ;       loops here - the correct one is selected based on the low-order bits        ;[E121]
                                ;       of the starting local buffer address. Each loop has two entry points,      ;[E121]
                                ;       so a 16-way branch is actually taken.                                       ;[E121]
                                ;
                                ;       Entry into this loop is via the PAR table in upper memory. The 16 PAR entries ;[E121]
                                ;       are all of the form:                                                        ;[E121]
                                ;
                                ;       SUB     #8.,UBAR                %JMP    BRnnLP                              ;[E121]
                                ;                                                                                   ;[E121]
                                ;                                                                                   ;[E121]
005672  031442  140422  006065  BR00LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005673  030145  017507  106052  BR00LX: ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005674  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005675  076501  163015  015677          XOR\L   (BCAIS),R1,BUF          %JCDONE BR08LP  ; Read last wd, br if ovlp BI rd done;[E121]
005676  033747  160010  126201  BIRD08: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005677  031442  140422  006065  BR08LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005700  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005701  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005702  076501  163015  015672          XOR\L   (BCAIS),R1,BUF          %JCDONE BR00LP  ; Read last wd, br if ovlp BI rd done;[E121]
005703  033747  160010  126201  BIRD00: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005704  031442  140422  105673          DEC     R2              @GORD   %JNMERR BR00LX  ; decr owd ct, start ovlp rd, br if err;[E121]
005705  013440  000000  116065          NOP                             %JMP    BI.UDX  ; Exit if BI error             ;[E121]
                                                                                                                    ;[E121]
005706  031442  140422  006065  BR01LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005707  030145  017507  106052  BR01LX: ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005710  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005711  076501  163015  015713          XOR\L   (BCAIS),R1,BUF          %JCDONE BR09LP  ; Read last wd, br if ovlp BI rd done;[E121]
005712  033747  160010  126201  BIRD09: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005713  031442  140422  006065  BR09LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005714  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005715  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005716  010545  007007  010000          ADD     #7,UBAR\N,BAR                           ; Block ovflo here - update BAR ;[E121]
005717  076501  163015  005706          XOR\L   (BCAIS),R1,BUF          %JCDONE BR01LP  ; Read last wd, br if ovlp BI rd done;[E121]
005720  033747  160010  126201  BIRD01: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005721  031442  140422  115707          DEC     R2              @GORD   %JNMERR BR01LX  ; decr owd ct, start ovlp rd, br if err;[E121]
005722  013440  000000  116065          NOP                             %JMP    BI.UDX  ; Exit if BI error             ;[E121]
                                                                                                                    ;[E121]
005723  031442  140422  006065  BR02LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005724  030145  017507  106052  BR02LX: ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005725  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005726  076501  163015  005730          XOR\L   (BCAIS),R1,BUF          %JCDONE BR10LP  ; Read last wd, br if ovlp BI rd done;[E121]
005727  033747  160010  126201  BIRD10: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005730  031442  140422  006065  BR10LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005731  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd    ;[E121]
005732  076501  003715  126070          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.5W  ; Read 6 words into local buffer;[E121]
```



```
005733  010545  007006  126074          ADD     #6,UBAR\N,BAR                   %CALL   BIR.1W  ; Handle block ovflo, read 1 wd ;[E121]
005734  076501  163015  015723          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR02LP  ; Read last wd, br if ovlp BI rd done;[E121]
005735  033747  160010  126201  BIRD02: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005736  031442  140422  105724          DEC     R2              @GORD           %JNMERR BR02LX  ; decr owd ct, start ovlp rd, br if err;[E121]
005737  013440  000000  116065          NOP                                     %JMP    BI.UDX  ; Exit if BI error             ;[E121]
                                                                                                                            ;[E121]
                                                                                                                            ;[E121]
005740  031442  140422  006065  BR03LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005741  030145  017507  106052  BR03LX: ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
005742  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005743  076501  163015  015745          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR11LP  ; Read last wd, br if ovlp BI rd done;[E121]
005744  033747  160010  126201  BIRD11: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005745  031442  140422  006065  BR11LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005746  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
005747  076501  003715  136071          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.4W  ; Read 5 words into local buffer;[E121]
005750  010545  007005  136073          ADD     #5,UBAR\N,BAR                   %CALL   BIR.2W  ; Handle block ovflo, read 2 wds;[E121]
005751  076501  163015  015740          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR03LP  ; Read last wd, br if ovlp BI rd done;[E121]
005752  033747  160010  126201  BIRD03: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005753  031442  140422  105741          DEC     R2              @GORD           %JNMERR BR03LX  ; decr owd ct, start ovlp rd, br if err;[E121]
005754  013440  000000  116065          NOP                                     %JMP    BI.UDX  ; Exit if BI error             ;[E121]
                                                                                                                            ;[E121]
                                                                                                                            ;[E121]
005755  031442  140422  006065  BR04LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005756  030145  017507  106052  BR04LX: ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
005757  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005760  076501  163015  015762          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR12LP  ; Read last wd, br if ovlp BI rd done;[E121]
005761  033747  160010  126201  BIRD12: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005762  031442  140422  006065  BR12LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005763  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
005764  076501  003715  136072          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.3W  ; Read 4 words into local buffer;[E121]
005765  010545  007004  136072          ADD     #4,UBAR\N,BAR                   %CALL   BIR.3W  ; Handle block ovflo, read 3 wds;[E121]
005766  076501  163015  005755          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR04LP  ; Read last wd, br if ovlp BI rd done;[E121]
005767  033747  160010  126201  BIRD04: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005770  031442  140422  105756          DEC     R2              @GORD           %JNMERR BR04LX  ; decr owd ct, start ovlp rd, br if err;[E121]
005771  013440  000000  116065          NOP                                     %JMP    BI.UDX  ; Exit if BI error             ;[E121]
                                                                                                                            ;[E121]
                                                                                                                            ;[E121]
005772  031442  140422  006065  BR05LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
005773  030145  017507  106052  BR05LX: ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
005774  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
005775  076501  163015  005777          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR13LP  ; Read last wd, br if ovlp BI rd done;[E121]
005776  033747  160010  126201  BIRD13: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
005777  031442  140422  006065  BR13LP: DEC     R2              @GORD           %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
006000  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST          %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                                ; br out if 2d-from-last owd    ;[E121]
006001  076501  003715  126073          XOR\L   (BCAIS),R1,BUF  @RDNXT          %CALL   BIR.2W  ; Read 3 words into local buffer;[E121]
006002  010545  007003  126071          ADD     #3,UBAR\N,BAR                   %CALL   BIR.4W  ; Handle block ovflo, read 4 wds;[E121]
006003  076501  163015  005772          XOR\L   (BCAIS),R1,BUF                  %JCDONE BR05LP  ; Read last wd, br if ovlp BI rd done;[E121]
006004  033747  160010  126201  BIRD05: MOV     #8.,R7                          %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
```

```
006005  031442  140422  115773          DEC     R2              @GORD   %JNMERR BRO5LX  ; decr owd ct, start ovlp rd, br if err;[E121]
006006  013440  000000  116065          NOP                             %JMP    BI.UDX  ; Exit if BI error                      ;[E121]
                                                                                                                                ;[E121]
                                                                                                                                ;[E121]
006007  031442  140422  006065  BRO6LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
006010  030145  017507  106052  BRO6LX: ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd     ;[E121]
006011  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
006012  076501  163015  016014          XOR\L   (BCAIS),R1,BUF          %JCDONE BR14LP  ; Read last wd, br if ovlp BI rd done;[E121]
006013  033747  160010  126201  BIRD14: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
006014  031442  140422  006065  BR14LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
006015  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd     ;[E121]
006016  076501  003715  136074          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.1W  ; Read 2 words into local buffer;[E121]
006017  010545  007002  126070          ADD     #2,UBAR\N,BAR           %CALL   BIR.5W  ; Handle block ovflo, read 5 wds;[E121]
006020  076501  163015  006007          XOR\L   (BCAIS),R1,BUF          %JCDONE BRO6LP  ; Read last wd, br if ovlp BI rd done;[E121]
006021  033747  160010  126201  BIRD06: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
006022  031442  140422  106010          DEC     R2              @GORD   %JNMERR BRO6LX  ; decr owd ct, start ovlp rd, br if err;[E121]
006023  013440  000000  116065          NOP                             %JMP    BI.UDX  ; Exit if BI error                      ;[E121]
                                                                                                                                ;[E121]
                                                                                                                                ;[E121]
006024  031442  140422  006065  BRO7LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
006025  030145  017507  106052  BRO7LX: ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd     ;[E121]
006026  076501  003715  126067          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.6W  ; Read 7 words into local buffer;[E121]
006027  076501  163015  006031          XOR\L   (BCAIS),R1,BUF          %JCDONE BR15LP  ; Read last wd, br if ovlp BI rd done;[E121]
006030  033747  160010  126201  BIRD15: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
006031  031442  140422  006065  BR15LP: DEC     R2              @GORD   %JMERR  BI.UDX  ; decr owd ct, start ovlp rd, br if err;[E121]
006032  030145  017507  106052          ADD     R7,UBAR,BAR     @RDFST  %JZRO   BIR.L2  ; upd buf adr, init BCAI rd ptr,;[E121]
                                                                                        ; br out if 2d-from-last owd     ;[E121]
006033  076501  003715  010000          XOR\L   (BCAIS),R1,BUF  @RDNXT                  ; Read 1 word into local buffer ;[E121]
006034  010545  007001  126067          ADD     #1,UBAR\N,BAR           %CALL   BIR.6W  ; Handle block ovflo, read 6 wd ;[E121]
006035  076501  163015  016024          XOR\L   (BCAIS),R1,BUF          %JCDONE BR07LP  ; Read last wd, br if ovlp BI rd done;[E121]
006036  033747  160010  126201  BIRD07: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for ovlp rd to finish;[E121]
006037  031442  140422  106025          DEC     R2              @GORD   %JNMERR BRO7LX  ; decr owd ct, start ovlp rd, br if err;[E121]
006040  013440  000000  116065          NOP                             %JMP    BI.UDX  ; Exit if BI error                      ;[E121]
                                .page                                                                                           ;[E121]
```



```
                                ;       This read loop is used at the tail end of the ultra-optimized loop                      ;[E121]
                                                                                                                                ;[E121]
006041  014145  040507  106053  BIR.LP: BIT     R7,UBAR         @RDFST  %JNLSB  BIR.EV  ; branch if UBAR even, ck opt case;[E121]
006042  076501  013715  116047  BIR.OD: XOR\L   (BCAIS),R1,BUF  @RDNXT  %JZRO   BIR.NI  ; UBAR odd - move 1st wd to mem ;[E121]
006043  010545  007001  126073          ADD     #1,UBAR\N,BAR           %CALL   BIR.2W  ; move 2d and 3d wds to mem     ;[E121]
006044  010545  007003  136073          ADD     #3,UBAR\N,BAR           %CALL   BIR.2W  ; move 4th and 5th wds to mem   ;[E121]
006045  010545  007005  136073          ADD     #5,UBAR\N,BAR           %CALL   BIR.2W  ; move 6th and 7th wds to mem   ;[E121]
006046  010545  007007  106060          ADD     #7,UBAR\N,BAR           %JMP    BIR.CM  ; go to common code for 8th wd  ;[E121]
006047  076501  003715  136074  BIR.NI: XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.1W  ; opt case - move 2d & 3d wds to mem;[E121]
006050  076501  003715  126073          XOR\L   (BCAIS),R1,BUF  @RDNXT  %CALL   BIR.2W  ; move 4th, 5th, 6th wds to mem ;[E121]
006051  076501  003715  106060          XOR\L   (BCAIS),R1,BUF  @RDNXT  %JMP    BIR.CM  ; move 7th wd, go to common code;[E121]
                                                                                                                                ;[E121]
                                ;       re-entry from ultra-optimized read code at BIR.L2                                       ;[E121]
                                                                                                                                ;[E121]
006052  014145  040007  016042  BIR.L2: BIT     R7,UBAR                         %JLSB   BIR.OD  ; branch if UBAR odd, ck opt case;[E121]
006053  076501  013715  116047  BIR.EV: XOR\L   (BCAIS),R1,BUF  @RDNXT  %JZRO   BIR.NI  ; UBAR even - move 1st wd to mem;[E121]
006054  076501  003715  010000          XOR\L   (BCAIS),R1,BUF  @RDNXT                  ; move 2d wd to mem             ;[E121]
006055  010545  007002  126073          ADD     #2,UBAR\N,BAR           %CALL   BIR.2W  ; move 3d and 4th wds to mem    ;[E121]
006056  010545  007004  126073          ADD     #4,UBAR\N,BAR           %CALL   BIR.2W  ; move 5th and 6th wds to mem   ;[E121]
006057  010545  007006  126074          ADD     #6,UBAR\N,BAR           %CALL   BIR.1W  ; move 7th wd to mem            ;[E121]
006060  076501  163015  006062  BIR.CM: XOR\L   (BCAIS),R1,BUF          %JCDONE BIR.QK  ; move 8th wd, br if BI rd done ;[E121]
006061  033747  160010  126201  BIR.LW: MOV     #8.,R7                  %CNCDONE BI.TO  ; R7=8, wait for BI rd to complete;[E121]
006062  031442  140002  056065  BIR.QK: DEC\F   R2                      %JMERR  BI.UDX  ; dec octawd ct, br if BI err   ;[E121]
006063  030145  027007  006041          ADD     R7,UBAR,BAR             %JNCRY  BIR.LP  ; if 1st time, reenter loop to dump;[E121]
                                                                                        ; last octwd without new BI read;[E121]
006064  034540  000007  106075          AND     #7,R0                   %JMP    BIR.EX  ; reduce WC, check partial owd  ;[E121]
                                                                                                                                ;[E121]
006065  013440  000000  126201  BI.UDX: NOP                             %CALL   BI.TO   ; wait for read to finish (if any);[E121]
006066  034540  000007  116303          AND     #7,R0                   %JMP    BI.UDO  ; reduce WC to partial owd      ;[E121]
                                                                                        ; and adjust addr/count to error point;[E121]
                                                                                                                                ;[E121]
                                ; routines to move one to six words from BCAI into buffer memory                               ;[E121]
                                                                                                                                ;[E121]
006067  076501  003715  010000  BIR.6W: XOR\L   (BCAIS),R1,BUF  @RDNXT                  ;move a word                    ;[E121]
006070  076501  003715  010000  BIR.5W: XOR\L   (BCAIS),R1,BUF  @RDNXT                  ;move a word                    ;[E121]
006071  076501  003715  010000  BIR.4W: XOR\L   (BCAIS),R1,BUF  @RDNXT                  ;move a word                    ;[E121]
006072  076501  003715  010000  BIR.3W: XOR\L   (BCAIS),R1,BUF  @RDNXT                  ;move a word                    ;[E121]
006073  076501  003715  010000  BIR.2W: XOR\L   (BCAIS),R1,BUF  @RDNXT                  ;move a word                    ;[E121]
006074  076501  003715  137777  BIR.1W: XOR\L   (BCAIS),R1,BUF  @RDNXT  %RTN            ;move a word and exit           ;[E121]
                                                                                                                                ;[E121]
                                                                                                                                ;[E121]
                                ; IF HERE, THEN < OCTAWORD TO XFER AND ON OCTAWORD ALIGNMENT                                    ;[E121]
006075  004240  010000  106231  BIR.EX: CLR     Q                       %JZRO   BI.EX   ; IF NO MORE WORDS/EXIT         ;[E121]
                                ;
                                ;       GET MORE LONGWORDS -- I LOVE EM!
                                ;
                                ;       LAST OCTAWORD; DON'T XFER FULL OCTAWORD
                                ;
006076  013440  005566  000000  BIR.X2: MOV     IUAR,BCAID      @LADD                   ;LOAD LO ADDRESS
006077  103543  000300  010000          BIS     #0W,R3,Q                ; LOAD HI ADDRESS BITS          ;[E121]
006100  013240  005760  010000          MOV     Q,BCAID         @HADD                                   ;[E121]
006101  033747  000001  010000          MOV     #BIRDCM,R7              ; LOAD COMMAND                  ;[E121]
006102  013440  005547  000000          MOV     R7,BCAID        @LCOM   ; AND GO READ                   ;[E121]
```

```
006103  013440  007425  010000          MOV     UBAR,BAR        @GORD                           ;                               ;[E121]
006104  013440  000000  126201          NOP                                     %CALL   BI.TO   ; AND TIME OUT THE READ         ;[E121]
006105  013440  140500  006210          NOP                     @RDFST  %JMERR  BI.ERR  ; IF BUS ERROR, EXIT                   ;[E121]
006106  076501  003715  010000  10$:     XOR\L   (BCAIS),R1,BUF  @RDNXT                  ; GET DATA                      ;[E121]
006107  130445  007005  010000          INC     UBAR,UBAR,BAR                           ; INCREMENT UBAR                ;[E121]
006110  031440  000000  000000          DEC     R0,R0                                   ; IF NOT DONE                   ;[E121]
006111  030546  010002  016106          ADD     #2,IUAR                         %JNZRO  10$     ;  LOOP/INCR ADDR (NO CARRY)    ;[E121]
006112  004240  000000  116231  BIREOK: CLR     Q                               %JMP    BI.EX   ; THEN EXIT                     ;[E121]
                                .PAGE
```



```
                                ; IF HERE THEN LESS THAN AN OCTAWORD TO XFER                                                    ;[E121]
                                                                                                                                ;[E121]
006113  053450  010006  116076  BIR.NO: MOV\R   IUAR,R10                      %JZRO   BIR.X2  ; ON OCTAWORD BOUNDARY,         ;[E121]
                                                                                                ; < OCTAWORD                    ;[E121]
006114  034550  000007  010000  BIR.SO: AND     #7,R10                                          ; R10 = OFFSET WITHIN OCTAWORD  ;[E121]
006115  132550  000010  000000          RSUB    #8.,R10                                         ; R10 = WDS TO END OF OCTAWORD  ;[E121]
006116  015546  005003  010000          BIC     #3,IUAR\N,BCAID                                 ; LOAD LO ADDRESS               ;[E121]
006117  112150  000560  010000          CMP     R0,R10          @LADD                           ;                               ;[E121]
006120  033470  030000  146121          MOV\T   R0,R10                          %TNEG           ; REDUCE COUNT TO STAY IN OWORD ;[E121]
006121  113543  005300  000000          BIS     #0W,R3\N,BCAID                                  ; LOAD HI ADDRESS BITS          ;[E121]
006122  013440  000760  010000          NOP                     @HADD                           ;                               ;[E121]
006123  003740  000001  000000          MOV     #BIRDCM,Q                                       ; LOAD COMMAND                  ;[E121]
006124  013240  005540  010000          MOV     Q,BCAID         @LCOM                           ;                               ;[E121]
006125  013440  007425  010000          MOV     UBAR,BAR        @GORD                           ; START READ                    ;[E121]
006126  033747  000002  136201          MOV     #2,R7                           %CALL   BI.TO   ; WAIT FOR READ TO COMPLETE     ;[E121]
006127  014146  140507  006210          BIT     R7,IUAR         @RDFST  %JMERR  BI.ERR  ; IF BUS ERROR, EXIT                   ;[E121]
006130  131140  010010  106132          SUB     R10,R0                          %JZRO   10$     ; ADJUST FINAL WC / BR IF EVEN WD;[E121]
006131  013440  000700  010000          NOP                     @RDNXT                          ; SKIP 1ST WD IN BCAI BUF       ;[E121]
                                                                                                                                ;[E121]
006132  076501  003715  010000  10$:     XOR\L   (BCAIS),R1,BUF  @RDNXT                  ; READ A WORD FROM BCAI BUF     ;[E121]
006133  130445  007005  010000          INC     UBAR,UBAR,BAR                           ; BUMP INTERNAL BUF ADR         ;[E121]
006134  031450  000010  000000          DEC     R10                                     ; DECR WD CT FOR THIS XFER      ;[E121]
006135  030546  010002  006132          ADD     #2,IUAR                         %JNZRO  10$     ; BUMP BI ADDR, BR IF MORE      ;[E121]
006136  130463  020003  146137          INC\T   R3                              %TCRY           ; ONLY NEED TO CK OVFLO AT END OF LOOP;[E121]
006137  013440  000000  000000          TST     R0                                              ; ANY WORDS LEFT TO XFER?       ;[E121]
006140  013440  010000  005643          NOP                                     %JNZRO  BI.RLP  ; IF SO, GO BACK TO XFER THEM   ;[E121]
006141  004240  000000  116231          CLR     Q                               %JMP    BI.EX   ; ELSE, EXIT                    ;[E121]
                                .PAGE
```

```
                                        ;                                                                                   ;[E121]
                                        ;       FOR DIAGNOSTIC MODE BI READ - ALLOWS WORD COUNTS IN EXCESS OF 256.           ;[E121]
                                        ;                                                                                   ;[E121]
006142  033747  000001  010000  DM.RD:  MOV     #BIRDCM,R7                                  ; COMMAND IN R7                 ;[E121]
006143  133543  000300  010000          BIS     #0W,R3                                                                      ;[E121]
006144  013440  005566  000000  DM.RLP: MOV     IUAR,BCAID      @LADD                       ; LOAD LO ADDRESS               ;[E121]
006145  013440  005763  010000          MOV     R3,BCAID        @HADD                       ; LOAD HI ADDRESS               ;[E121]
006146  013440  005547  000000          MOV     R7,BCAID        @LCOM                       ; LOAD COMMAND IN R7            ;[E121]
006147  013440  007425  010000          MOV     UBAR,BAR        @GORD                                                       ;[E121]
006150  030546  000020  010000          ADD     #16.,IUAR                                   ; INCREMENT ADDRESS             ;[E121]
006151  131540  020010  016153          SUB     #8.,R0                      %JNCRY  10$     ; DECREMENT WORD COUNT          ;[E121]
006152  130443  000003  000000          INC     R3                                          ; INCREMENT HI ADDRESS IF CARRY SET ;[E121]
006153  013440  000000  126201  10$:    NOP                                 %CALL   BI.TO   ; GO READ                      ;[E121]
006154  130445  140505  056210          INC\F   UBAR            @RDFST      %JMERR  BI.ERR  ; IF ERROR, SET Q, EXIT         ;[E121]
006155  076501  003715  010000          XOR\L   (BCAIS),R1,BUF  @RDNXT                      ; READ THE DATA                 ;[E121]
006156  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006157  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006160  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006161  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006162  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006163  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006164  120445  007005  126074          INC     UBAR\0,UBAR,BAR             %CALL   BIR.1W                                  ;[E121]
006165  031442  000002  000000          DEC     R2                                          ; DECREMENT OCTAWORD COUNT      ;[E121]
006166  135543  010300  046144          BIC\F   #0W,R3                      %JNZRO  DM.RLP  ; AND LOOP                      ;[E121]
006167  013440  000000  105075          TST     R0                          %JMP    BIR.EX  ; CHECK PARTIAL OWD AND EXIT    ;[E121]
                                        .page                                                                               ;[E121]
```



```
                                        ;*************************************************************************          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;  CHKBDR  check for page boundary crossing.                            *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;*************************************************************************          ;[E121]
                                        ;  ASSumptions and interface:                                          *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;       1.  Since the word count is <= 256,                             *          ;[E121]
                                        ;           and the page size is 512 bytes, there is                    *          ;[E121]
                                        ;           at most one page boundary in any transfer                   *          ;[E121]
                                        ;           (DM mode is an exception handled seperately)                *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;       2.  Inputs                                                      *          ;[E121]
                                        ;               r0      : word count                                    *          ;[E121]
                                        ;               iuar    : low 16 bits of BI address                     *          ;[E121]
                                        ;               r3      : high bits of BI addr                          *          ;[E121]
                                        ;               r7      : IUAR negated and shifted right 1 bit          *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;       3.  Outputs                                                     *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;               BCAI next pg reg = next pg addr if page ovflo           *          ;[E121]
                                        ;               r7 is destroyed, all other regs are preserved           *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;       4.  Subroutines used:  none.                                    *          ;[E121]
                                        ;                                                                       *          ;[E121]
                                        ;*************************************************************************          ;[E121]
006170  034547  000377  000000  CHKBDR: AND     #377,R7                                     ; R7 now has wds left in page   ;[E121]
006171  131147  010000  137777          SUB     R0,R7                       %RZRO           ; ret if buf page aligned,      ;[E121]
                                                                                            ; else check pg ovflo           ;[E121]
006172  133747  030002  067777          MOV\F   #1000,R7                    %RNMSB          ; return if no pg ovflo         ;[E121]
006173  030147  000006  000000          ADD     IUAR,R7                                     ; Q = IUAR + #BYTES             ;[E121]
006174  134547  025376  016176          AND     #177000,R7,BCAID            %JNCRY  10$     ; LOAD LOW NEXT PAGE ADDRESS    ;[E121]
006175  130447  000443  116177          INC     R3,R7           @LNPAG      %JMP    20$     ; Increment hi adr if carry     ;[E121]
006176  033447  000443  000000  10$:    MOV     R3,R7           @LNPAG                      ; copy hi addr for BI reg       ;[E121]
006177  133547  000300  000000  20$:    BIS     #0W,R7                                      ; SET FOR OCTAWORD              ;[E121]
006200  013440  005767  127777          MOV     R7,BCAID        @HADD       %RTN            ; SET UP HI ADDRESS BITS        ;[E121]
                                        .page
```

```
                                ;*****************************************************************************
                                ;                                                                           *
                                ; Timeout routine.  This routine gets called only after an initial         *                ;[E121]
                                ;                   test for completion of a BI read/write has been       *                ;[E121]
                                ;                   unsuccessfully attempted. Bus timeouts are ignored.   *                ;[E121]
                                ;                   NOTE: do not start the BI I/O on the same cycle as    *                ;[E121]
                                ;                   the call to BI.TO, as CDONE may be erroneously set.   *                ;[E121]
                                ;                                                                           *
                                ;*****************************************************************************

006201  013440  160000  027777  BI.TO:  NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006202  013440  160000  027777         NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006203  013440  160000  027777         NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006204  013440  160000  027777         NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006205  013440  160000  027777         NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006206  013440  160000  027777         NOP                             %RCDONE          ; EXIT IF COMMAND COMPLETE AND ERROR FREE;[E121]
006207  013440  000000  116201         NOP                             %JMP     BI.TO   ; KEEP LOOPING                            ;[E121]
                                .page
```



```
                                ;       Various exit routines from BI transfer code                                 ;[E121]
                                                                                                                    ;[E121]
006210  013440  000400  010000  BI.ERR: NOP                     @SUPF                   ; IF BUS ERROR, SET UPF   ;[E121]
006211  013440  000660  000000         NOP                     @RBCAI                                            ;[E121]
006212  013740  116220  046235         MOV\F   #<B.LED+B.NRTY>,BREG    %JSTOP  BI.STO  ; [mjt06] STAY HERE IF STOP/CLEAR ERRORS;[E121]
006213  013740  006221  010000         MOV     #<B.LED+B.NRTY+B.NABO>,BREG             ; [mjt06] RESET BREG      ;[E121]
006214  033447  000000  000000         MOV     R0,R7                                   ; [mjt06] SAVE REGS       ;[E121]
006215  033450  000001  010000         MOV     R1,R10                                  ;                         ;[E121]
006216  033442  000005  000000         MOV     UBAR,R2                                 ;                         ;[E121]
                                                                                                                    ;[E121]
                                ;       read the bus error register for analyzer scoping purposes [mjt06]           ;[E121]
                                ;       Bug fix - can't call RDCMD here, we will exceed subr depth! [r1]            ;[E121]
                                                                                                                    ;[E121]
006217  033740  000010  130265         MOV     #BIBER,R0               %CALL   BRGADR  ;BIIC REGISTER ADDRESS = Bus error reg;[E121]
006220  033740  000001  000000         MOV     #BIRDCM,R0                              ;GET READ COMMAND         ;[E121]
006221  033440  115540  006237         MOV     R0,,BCAID @LCOM         %JSTOP  BI.STP  ;JUMP IF BI STOPPED / LOAD COMMAND      [mr1001];[E1
006222  013440  000420  130323         NOP             @GORD           %CALL   WTCMDD  ;DO THE READ /            ;[E121]
                                                                                        ; GO WAIT FOR CMD COMPLETE & ERROR FREE;[E121]
006223  033440  000507  000000         MOV     R7,R0                   @RDFST          ;READ 1ST BUFFER, RESTORE R0    ;[E121]
006224  013700  000715  010000         MOV     (BCAIS),R0\N            @RDNXT          ;GET DATA FROM 1ST BCAI BUFFER /;[E121]
                                                                                        ; READ NEXT BUFFER.       ;[E121]
006225  013700  000015  000000         MOV     (BCAIS),R0\N                            ;GET DATA FROM NEXT BCAI BUFFER /;[E121]
006226  033441  000010  010000         MOV     R10,R1                                  ; [mjt06] RESTORE REGS    ;[E121]
006227  033445  000002  000000         MOV     R2,UBAR                                 ;                         ;[E121]
006230  003740  000002  000000         MOV     #2,Q                                    ; set error code          ;[E121]
006231  013740  007044  125601  BI.EX:  MOV     #CNVTV3,BAR             %CALL   S.LDR7  ; RESTORE R7              ;[E121]
006232  013740  007045  135602         MOV     #CNVTV4,BAR             %CALL   S.LD10  ; RESTORE R10             ;[E121]
006233  013740  007262  135622         MOV     #SAVUAR,BAR             %CALL   S.STR3  ; SAVE UPDATED SAVUAR     ;[E121]
006234  013240  000000  127777         TST     Q                       %RET            ; RETURN STATUS           ;[E121]

006235  013740  007034  000000  BI.STO: MOV     #FAILUR,BAR                             ; [mjt06] save stop command in last fail
006236  013740  003146  000000         MOV     #ER.STP,BUF                             ; [mjt06]
006237  013740  006220  000000  BI.STP: MOV     #<B.LED+B.NRTY>,BREG                    ; [mjt06] clear abort
006240  013740  006221  010000         MOV     #<B.LED+B.NRTY+B.NABO>,BREG
006241  113740  004020  106237         MOV     #<LED1>,UCRD            %JMP    BI.STP  ; [mjt06] hang forever
                                                                                        ; because we got the stop command
                                .page
```

```
                        ;*****************************************************************************
                        ;                                                                            *
                        ;       BI.WR  BI write routine                                              *
                        ;                                                                            *
                        ;*****************************************************************************
                        ;                                                                            *
                        ;  1.  Inputs (same as unb.wr):                                             *
                        ;               ubar    :  starting internal buffer ptr                      *          ;[E121]
                        ;               iuar,uar:  low 16 bits of external bus address               *          ;[E121]
                        ;                          for start of transfer.  Identical copies.         *          ;[E121]
                        ;               ucrd    :  holds high order bits of external bus addr.       *          ;[E121]
                        ;               r0      :  word count for the transfer                       *          ;[E121]
                        ;               savuar  :  holds high order bits of bus address.             *          ;[E121]
                        ;                                                                            *
                        ;   2. Outputs (same as unb.wr):                                            *
                        ;               q       :  error code                                        *
                        ;                          If r0 <> 0,                                       *
                        ;                          q = 1 means nxm error (time-out)                  *
                        ;               upf     : (ioc flag) set means timeout, not                  *
                        ;                             set means parity                               *
                        ;               r0      :  number of words not transfered                    *
                        ;                          On success, transfer will be 0.                   *
                        ;               r1      :  EDC                                               *
                        ;                                                                            *
                        ;  3.  Side-effects:                                                        *
                        ;               iuar, uar are advanced.                                      *
                        ;               ubar,bar  are advanced.                                      *
                        ;               Host data in buffer pointed to by ubar on entry.             *
                        ;                                                                            *
                        ;               r2 and r3 are not preserved.                                 *
                        ;               r7 used as scratch but original contents saved and           *
                        ;               restored                                                     *
                        ;                                                                            *
                        ;  4.  Entry points:                                                        *
                        ;               a.  the usual entry is qb.wr or bi.wr.                       *          ;[E121]
                        ;               b.  unb.wr is included for consistency                       *
                        ;                   to insure that calls from common unibus                  *
                        ;                   and qbus call use the correct bus writing routine.       *
                        ;               c.  WT.DIAG is a diagnostic entry point and the only         *
                        ;                   difference is that the last fail code is not set         *
                        ;                   (edc is set before the call).                            *
                        ;               d.  UN.BWC is used in order not to reset the EDC residue     *          ;[E121]
                        ;                                                                            *
                        ;*****************************************************************************
                        .page
```



```
                        ;
                        ;       ON OCTAWORD BOUNDARY? -> NO -> THEN XFER UNTIL ON OCTAWORD BOUNDARY
                        ;               .                                .
                        ;               YES                              .
                        ;               .                                .
                        ;               V                                .
                        ;               MORE OCTAWORDS XFER?<------------+<---------------+
                        ;               .                   .                              .
                        ;               NO                  YES-----)-----XFER OCTAWORD-----+
                        ;               .
                        ;               V
                        ;               ANY DATA LEFT TO XFER?->YES---->THEN XFER REST
                        ;               .                                .
                        ;               NO                               .
                        ;               .<---------------------------------------+
                        ;               V
                        ;               EXIT
                        ;
                        UNB.WR:                                 ; FOR UDA CALL COMPATIBILITY
                        QB.WR:                                  ; FOR QDA CALL COMPATIBILITY
                        BI.WR:
006242  033741  000105  010000          MOV     #EDSEED,R1      ; R1 WILL CONTAIN EDC THROUGHOUT
006243  013740  007034  000000  UN.BWC: MOV     #FAILUR,BAR                     ; STORE FAILURE CODE IF
006244  013740  003013  010000          MOV     #ER.MST,BUF                     ;  BI STOP COMMAND OCCURS
006245  013740  007044  125625  WT.DIAG: MOV    #CNVTV3,BAR             %CALL   S.STR7  ; SAVE R7 AND R10
006246                          ASSUME  <CNVTV4&17>,EQ,<CNVTV3&17>+1
006246  033450  003010  010000          MOV     R10,R10,BUF                                                     ;[E121]
006247  013740  007262  125576          MOV     #SAVUAR,BAR             %CALL   S.LDR3  ; R3 = IUAR EXTENSION   ;[E121]
                                                                                                                ;[E121]
006250  013740  116221  006235  BIW.LQ: MOV     #<B.LED+B.NABO+B.NRTY>,BREG %JSTOP BI.STO ; DIE IF BI STOP ISSUED ;[E121]
                                                                                ; SET SELF TEST OK, NO ABORT CMD, NO RETRY;[E121]
006251  053442  000660  000000          MOV\R   R0,R2           @RBCAI          ; R2 = # OF LONGWORDS           ;[E121]
006252  055542  010003  106112          BIC\R   #3,R2                   %JZRO   BIREOK  ; R2 = # OF QUADWORDS   ;[E121]
006253  155542  000140  000000          BIC\R   #60000,R2                               ; R2  # OF OCTAWORDS    ;[E121]
006254  014546  010016  106417          BIT     #16,IUAR                %JZRO   BIW.NO  ; BR IF NO OCTAWDS TO XFER ;[E121]
006255  053467  010006  046361          MOV\RT  IUAR,R7                 %JNZRO  BIW.SO  ; BR IF NOT ON OCTAWORD BOUNDARY;[E121]
006256  152447  000006  126170          NEG\R   IUAR,R7                 %CALL   CHKBDR  ; CAN XFER OWDS - CHECK FOR PG OVFLO;[E121]
006257  114544  000100  010000          BIT     #DMODE,RLL                      ; IF IN DIAG MODE, USE SPECIAL  ;[E121]
006260  073447  010000  016430          MOV\L   R0,R7                   %JNZRO  DM.WR   ; ROUTINE; R7 = BYTE COUNT      ;[E121]
006261  013440  005566  000000          MOV     IUAR,BCAID      @LADD                   ; LOAD LOW ADDR BITS    ;[E121]
006262  113543  005300  000000          BIS     #0W,R3\N,BCAID                  ; SET UP OCTAWORD XFER          ;[E121]
006263  013440  007765  000000          MOV     UBAR,BAR        @HADD           ; WITH HI ADDR BITS AND LOAD BAR;[E121]
006264  013740  005004  010000          MOV     #BIWRCM,BCAID                   ; LOAD BI WRITE COMMAND         ;[E121]
006265  035547  000017  010000          BIC     #17,R7                          ; assume successful xfer,       ;[E121]
006266  030146  000547  010000          ADD     R7,IUAR         @LCOM           ; and update BI addr            ;[E121]
006267  034540  020007  006271          AND     #7,R0                   %JNCRY  10$     ; and wd ct             ;[E121]
006270  130443  000003  000000          INC     R3                              ; (BI addr is double precision!);[E121]
006271  076501  005523  126312  10$:    XOR\L   (BUF),R1,BCAID  @WRFST  %CALL   BIW.5R  ; load first 6 words of octwd ;[E121]
006272  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT          ; load 7th word of octwd        ;[E121]
006273  130445  007005  010000          INC     UBAR,UBAR,BAR                                                   ;[E121]
006274  076501  005743  010000  BIW.LP: XOR\L   (BUF),R1,BCAID  @LBWR                  ; LOAD LAST WORD AND WRITE ONTO BI BUS;[E121]
006275  031442  000002  000000          DEC     R2                              ; dec octawd ct                 ;[E121]
006276  130445  017005  106325          INC     UBAR,UBAR,BAR           %JZRO   BIW.EX  ; adjust UBAR, br if last octawd;[E121]
006277  076501  005523  126312          XOR\L   (BUF),R1,BCAID  @WRFST  %CALL   BIW.5R  ; write first 6 words  ;[E121]
```

```
006300  076501  165723  016302          XOR\L   (BUF),R1,BCAID  @WRNXT  %JCDONE BIW.QK  ; write 7th wd, ck BI done      ;[E121]
006301  013440  160000  136201          NOP                             %CNCDONE BI.TO  ; wait for BI write            ;[E121]
006302  130445  147005  116274  BIW.QK: INC     UBAR,UBAR,BAR           %JNMERR BIW.LP  ; loop if no err               ;[E121]
                                                                                                                       ;[E121]
006303  073447  000002  000000  BI.UDO: MOV\L   R2,R7                                  ; err - adjust wd ct and BI addr;[E121]
006304  070147  000007  000000          ADD\L   R7,R7                                  ; to show correct stopping point;[E121]
006305  030140  000007  000000          ADD     R7,R0                                                                  ;[E121]
006306  030147  000007  010000          ADD     R7,R7                                                                  ;[E121]
006307  131146  000007  000000          SUB     R7,IUAR                                                                ;[E121]
006310  031463  020003  056311          DEC\T   R3                      %TNCRY                                         ;[E121]
006311  013440  000000  116210          NOP                             %JMP    BI.ERR  ;report bus error             ;[E121]
                                                                                                                       ;[E121]
                                        ; Subroutine to load 5 words into BCAI and bump UBAR 6 times                   ;[E121]
                                                                                                                       ;[E121]
006312  130445  007005  010000  BIW.5R: INC     UBAR,UBAR,BAR                                                          ;[E121]
006313  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT
006314  130445  007005  010000          INC     UBAR,UBAR,BAR
006315  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT
006316  130445  007005  010000          INC     UBAR,UBAR,BAR
006317  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT
006320  130445  007005  010000          INC     UBAR,UBAR,BAR
006321  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT
006322  130445  007005  010000          INC     UBAR,UBAR,BAR
006323  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT
006324  130445  007005  137777          INC     UBAR,UBAR,BAR           %RTN                                           ;[E121]
                              .page                                                                                    ;[E121]
```



```
                                                                                                                       ;[E121]
                                ;       Exit from octaword write loop                                                  ;[E121]
                                                                                                                       ;[E121]
006325  013440  000000  126201  BIW.EX: NOP                             %CALL   BI.TO   ; wait for last BI write       ;[E121]
006326  013440  140000  006210  BIW.TS: TST     R0                      %JMERR  BI.ERR  ; br if bus err                ;[E121]
006327  004260  010000  146231          CLR\T   Q                       %JZR0   BI.EX   ; exit if no partial octwd     ;[E121]
                                ;                                                                                      ;[E121]
                                ;       Come here when we are on a longword boundary and will not cross the            ;[E121]
                                ;       next octaword boundary - short transfers and the end of long ones.             ;[E121]
                                ;                                                                                      ;[E121]
                                                                                                                       ;[E121]
006330  111540  000002  000000  BIW.LS: SUB     #2,R0\N                                 ; CHECK FOR WC >= ONE LONGWD   ;[E121]
006331  033747  030004  146346          MOV\F   #BIWRCM,R7              %JMSB   20$     ; IF < 1 LWD, SKIP             ;[E121]
006332  013440  005547  000000          MOV     R7,BCAID        @LCOM                   ; LOAD BI WRITE COMMAND        ;[E121]
006333  013440  005566  000000          MOV     IUAR,BCAID      @LADD                   ; LOW ADDRESS LOADED           ;[E121]
006334  113543  005100  010000          BIS     #LW,R3\N,BCAID                          ; LOAD HI ADDRESS              ;[E121]
006335  013440  007765  000000          MOV     UBAR,BAR        @HADD                   ; LOAD BREG                    ;[E121]
006336  076501  005523  000000  10$:    XOR\L   (BUF),R1,BCAID  @WRFST                  ; WRITE FIRST                  ;[E121]
006337  130445  007005  010000          INC     UBAR,UBAR,BAR                           ; NEXT WORD                    ;[E121]
006340  076501  005743  010000          XOR\L   (BUF),R1,BCAID  @LBWR
006341  130445  007005  136201          INC     UBAR,UBAR,BAR           %CALL   BI.TO   ; WAIT FOR WRITE TO COMPLETE   ;[E121]
006342  030546  000004  010000          ADD     #4,IUAR                                 ; BUMP ADDRESS                 ;[E121]
006343  130463  020003  146344          INC\T   R3                      %TCRY                                          ;[E121]
006344  131540  140004  046210          SUB\F   #4,R0                   %JMERR  BI.ERR  ; EXIT IF ERR, ELSE DECR WC & CHECK FOR MORE;[E121]
006345  030540  030002  016336          ADD     #2,R0                   %JNMSB  10$     ; IF SO, BRANCH BACK           ;[E121]
                                                                                                                       ;[E121]
006346  033747  040007  146360  20$:    MOV\F   #WMCI,R7                %JNLSB  30$     ; IF NO ODD WD, EXIT           ;[E121]
006347  013440  005547  000000          MOV     R7,BCAID        @LCOM                   ; LOAD BI WRITE COMMAND        ;[E121]
006350  013440  005566  000000          MOV     IUAR,BCAID      @LADD                   ; LOW ADDRESS LOADED           ;[E121]
006351  103543  000100  000000          BIS     #LW,R3,Q                                                               ;[E121]
006352  013240  005760  010000          MOV     Q,BCAID         @HADD                   ; LOAD HI ADDRESS              ;[E121]
006353  013440  007685  010000          MOV     UBAR,BAR        @RBCAI                                                 ;[E121]
006354  076501  005523  000000          XOR\L   (BUF),R1,BCAID  @WRFST                                                 ;[E121]
006355  130445  007645  000000          INC     UBAR,UBAR,BAR   @GOWR                   ; START WRITE                  ;[E121]
006356  030546  000002  136201          ADD     #2,IUAR                 %CALL   BI.TO   ; BUMP ADR, WAIT FOR WRITE     ;[E121]
006357  034440  140000  056210          CLR\F   R0                      %JMERR  BI.ERR  ; EXIT IF ERR, ELSE CLEAR WC   ;[E121]
006360  004240  000000  116231  30$:    CLR     Q                       %JMP    BI.EX   ; AND EXIT                     ;[E121]
                                                                                                                       ;[E121]
                              .page                                                                                    ;[E121]
```

```
                                ; MORE THAN AN OCTAWORD TO XFER BUT NOT STARTING ON AN OCTAWORD BOUNDARY                ;[E121]
                                                                                                                       ;[E121]
006361  014546  000002  010000  BIW.S0: BIT     #2,IUAR                                  ; MASK NEEDED?                ;[E121]
006362  033747  010007  106367         MOV     #WMCI,R7                 %JZRO   BW.S01  ; IF NOT, BRANCH             ;[E121]
006363  013440  005547  126407         MOV     R7,BCAID        @LCOM    %CALL   BI.WOW  ; PERFORM ODD WORD WRITE      ;[E121]
006364  030546  000002  010000         ADD     #2,IUAR                                  ; BUMP BI ADDRESS             ;[E121]
006365  130463  020003  146366         INC\T   R3                       %TCRY                                       ;[E121]
006366  013440  140000  006210         NOP                              %JMERR  BI.ERR  ; IF ERROR, EXIT              ;[E121]
                                                                                                                       ;[E121]
                                ;       IUAR IS NOW ON A LONGWORD BOUNDARY                                             ;[E121]
                                                                                                                       ;[E121]
006367  152442  000006  000000  BW.S01: NEG\R   IUAR,R2                                                               ;[E121]
006370  054542  000006  000000         AND\R   #6,R2                                                                 ;[E121]
006371  033747  010004  156250         MOV\F   #BIWRCM,R7               %JZRO   BIW.LQ  ; R2 = # LONGWORDS            ;[E121]
006372  013440  005547  000000         MOV     R7,BCAID        @LCOM                    ; IF NOT, SKIP                ;[E121]
006373  013440  005566  000000         MOV     IUAR,BCAID      @LADD                    ; LOAD BI WRITE COMMAND       ;[E121]
006374  113543  005100  010000         BIS     #LW,R3\N,BCAID                           ; LOAD LOW ADDR               ;[E121]
006375  013440  007765  000000         MOV     UBAR,BAR        @HADD                                                 ;[E121]
006376  076501  005523  000000  15$:    XOR\L   (BUF),R1,BCAID  @WRFST                   ; LOAD HI ADDRESS & BAR       ;[E121]
006377  130445  007005  010000         INC     UBAR,UBAR,BAR                            ; WRITE FIRST
006400  076501  005743  010000         XOR\L   (BUF),R1,BCAID  @LBWR                    ; NEXT WORD
006401  130445  007005  136201         INC     UBAR,UBAR,BAR                    %CALL   BI.TO   ; WAIT FOR WRITE TO COMPLETE  ;[E121]
006402  030546  140004  046210         ADD\F   #4,IUAR                          %JMERR  BI.ERR  ; EXIT IF ERR, ELSE BUMP ADR  ;[E121]
006403  130463  020003  146404         INC\T   R3                               %TCRY                                 ;[E121]
006404  031442  000002  000000         DEC     R2                                       ; DONE?                       ;[E121]
006405  131540  010002  016376         SUB     #2,R0                            %JNZRO  15$     ; IF NOT, BRANCH      ;[E121]
006406  013440  000000  106250         NOP                                      %JMP    BIW.LQ                        ;[E121]
                                                                                                                       ;[E121]
                                ;       Perform odd word write - load address, load data, start write, and wait.       ;[E121]
                                ;       Side effects - UBAR/BAR incremented, R0 (word ct) decremented, EDC updated,    ;[E121]
                                ;       Q register destroyed.                                                          ;[E121]
                                                                                                                       ;[E121]
006407  005546  000003  000000  BI.WOW: BIC     #3,IUAR,Q                                ; CLEAR ODD WORD ALIGNMENT    ;[E121]
006410  013240  005560  000000         MOV     Q,BCAID         @LADD                                                 ;[E121]
006411  103543  000100  000000         BIS     #LW,R3,Q                                                              ;[E121]
006412  013240  005760  010000         MOV     Q,BCAID         @HADD                    ; LOAD HI ADDRESS             ;[E121]
006413  021340  007665  010000         DECB    UBAR\0,R0,BAR   @RBCAI                   ; DECR WC, LOAD BAR           ;[E121]
006414  076501  005463  000000         XOR\L   (BUF),R1,BCAID  @WRSND                   ; LOAD WORD                   ;[E121]
006415  130445  007645  000000         INC     UBAR,UBAR,BAR   @GOWR                    ; START WRITE                 ;[E121]
006416  013440  000000  116201         NOP                              %JMP    BI.TO   ; WAIT FOR COMPLETION & RETURN ;[E121]
                                .page                                                                                  ;[E121]
```



```
                                ; MUST BE < OCTAWORD TO BE HERE

006417  152447  010666  106330  BIW.N0: NEG\R   IUAR,R7          @RBCAI  %JZRO   BIW.LS  ; IF ON BOUNDARY, BRANCH       ;[E121]
006420  034547  000007  010000         AND     #7,R7                                    ; R7 = wds to octawd boundary  ;[E121]
006421  131147  000000  000000         SUB     R0,R7                                    ; DO WE CROSS AN OCTAWORD BOUNDARY?;[E121]
006422  014546  030002  116361         BIT     #2,IUAR                          %JMSB   BIW.S0  ; IF WE DO, BIW.S0 WILL HANDLE IT.;[E121]
                                                                                         ; ELSE TEST FOR ODD WORD      ;[E121]
006423  003740  010007  156330         MOV\F   #WMCI,Q                          %JZRO   BIW.LS  ; IF NOT, BIW.LS WILL HANDLE IT ;[E121]
006424  013240  005540  136407         MOV     Q,BCAID         @LCOM    %CALL   BI.WOW  ; WRITE ODD WORD              ;[E121]
006425  030546  000002  010000         ADD     #2,IUAR                                  ; BUMP BI ADDRESS             ;[E121]
006426  130443  020003  056326         INC\F   R3                               %JNCRY  BIW.TS  ; OTHERWISE BUMP ADDRESS      ;[E121]
006427  013440  000000  106326         NOP                                      %JMP    BIW.TS  ; AND LET BIW.LS DO THE REST  ;[E121]
                                .PAGE
```

```
                                ;                                                                                       ;[E121]
                                ;       FOR DIAGNOSTIC MODE BI WRITE - ALLOWS WORD COUNTS IN EXCESS OF 256              ;[E121]
                                ;                                                                                       ;[E121]
006430  013440  005566  000000  DM.WR:  MOV     IUAR,BCAID      @LADD                   ; LOAD ADDRESS                  ;[E121]
006431  113543  005300  000000          BIS     #OW,R3\N,BCAID                                                          ;[E121]
006432  013440  007765  000000          MOV     UBAR,BAR        @HADD                                                   ;[E121]
006433  033747  000004  010000          MOV     #BIWRCM,R7                                                              ;[E121]
006434  013440  005547  000000          MOV     R7,BCAID        @LCOM                   ; LOAD COMMAND                  ;[E121]
006435  076501  005523  126312          XOR\L   (BUF),R1,BCAID  @WRFST  %CALL   BIW.5R  ; LOAD FIRST WORD               ;[E121]
006436  076501  005723  010000          XOR\L   (BUF),R1,BCAID  @WRNXT                                                  ;[E121]
006437  130445  007005  010000          INC     UBAR,UBAR,BAR                                                           ;[E121]
006440  076501  005743  010000          XOR\L   (BUF),R1,BCAID  @LBWR                   ; LOAD LAST WORD AND WRITE ONTO BI BUS ;[E121]
006441  030546  000020  010000          ADD     #16.,IUAR                               ; INCREMENT ADDRESS             ;[E121]
006442  130463  020003  146443          INC\T   R3                              %TCRY   ; IF CARRY, INCREMENT HI ADDRESS BITS ;[E121]
006443  130445  000005  126201          INC     UBAR                            %CALL   BI.TO   ; ADJUST UBAR, TIMEOUT COMMAND  ;[E121]
006444  031442  140002  056210          DEC\F   R2                              %JMERR  BI.ERR  ; BR IF BUS ERROR, ELSE DECR OWD CT ;[E121]
006445  131540  010010  016430          SUB     #8.,R0                          %JNZRO  DM.WR   ; LOOP IF NOT DONE              ;[E121]
006446  004280  010000  146231          CLR\T   Q                               %JZRO   BI.EX   ; THEN EXIT IF ALL DONE         ;[E121]
006447  013440  000000  116330          NOP                                     %JMP    BIW.LS  ; ELSE, SEND LAST FEW DATA WORDS ;[E121]
                                                                                                ; (LESS THAN AN OCTAWORD)      ;[E121]
                                .PAGE
```



```
                                .SBTTL  QDA ECC CORRECTION ROUTINE
                                ;       05-AUG-83       QDA MICROCODE
                                ;       New Version with HSC algorithms - Beckman -
                                ;       Optimized Version - Version 3
                                ; MATT  01-JUL-83 CHANGES PER RICHY'S SPECIFICATION
                                ;
                                ; ROUTINE NAME:
                                ;       ECCC (ECC CORRECTION)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE PERFORMS THE HSC ECC CORRECTION ALGORITHM
                                ;       ON THE DATA BUFFER WITHIN THE SECTOR BUFFER POINTED TO BY R7.
                                ;
                                ; INPUTS:
                                ;       DMREG2          SECTOR SIZE IN WORDS
                                ;       R7              POINTER TO SECTOR BUFFER CONTROL AREA
                                ;       (BUF.BP+SECSZ+BUF.EC) = POINTER TO 12 WORDS OF PACKED ECC RESIDUES
                                ;
                                ; OUTPUTS:
                                ;       CONDITION CODE  ZRO   - ALL ECC CORRECTION MADE
                                ;       CONDITION CODE  NZRO  - DATA BUFFER NOT CORRECTABLE
                                ;       DMREGO                  NUMBER OF CORRECTIONS MADE
                                ;
                                ;;-

                                ;ECC CORRECTION ALGORITHM.

                                ;*********************************************************
                                ; Syndrome Computation
                                ;
                                ; There are 17 Syndrome values.Each is the remainder of the
                                ; division of two polynomials.The 17 residue values,treated as
                                ; coefficients of a polynomial is divided by each of the factors
                                ; of the Code Generation Polynomial.These factors are:
                                ;         -8      -7              0               7       8
                                ; (X+alpha  ),(X+alpha  ),...,(X+alpha ),...,(X+alpha  ),(X+alpha  )
                                ;                                  -8
                                ; The remainder of the division by (X+alpha  ) yields Syndrome (-8) etc.
                                ;
                                ; These syndromes turn out to be power series with one term for each
                                ; symbol in error. For v errors,at locations j(0) through j(v-1)
                                ; with values Y(1) through Y(v) Syndrome i has the following form:
                                ;
                                ;           (i)j(0)         (i)j(1)             (i)j(v-1)
                                ; S(i)= Alpha      Y(1) + Alpha      Y(2) +...+ Alpha       Y(v)
                                ;
                                ;*********************************************************

006450  013740  007172  135625  ECCC:   MOV     #SAVR7,BAR      %CALL   S.STR7  ; SAVE SECTOR BUFFER PTR
                                ; ***   UNPACK 12 WORDS OF PACKED RESIDUES INTO 18 WORDS - STARTING AT ELPN ***
                                ; ***   RESIDUES ARE UNPACKED LAST TO FIRST, I.E. 1ST RESIDUE IS AT ELPN ***
006451  010547  007002  135576          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR3  ; [16K]R3=DATA BUFFER POINTER
006452  013740  007002  135573          MOV     #DMREG2,BAR     %CALL   S.LDRO  ; [16K]RO=SECTOR SIZE
006453  030143  000000  010000          ADD     RO,R3                           ; [16K]ADD SECTOR SIZE TO R3
```

```
006454  030543  000014  000000          ADD     #<BUF.DL-401>,R3                ; [16K]FORM LAST RESIDUE ADDRESS
006455  033742  000056  136760          MOV     #ELPN+17.,R2    %CALL   GETRES  ; R2=ADDRESS OF LAST UNPACKED RESIDUE
006456  033746  000006  010000          MOV     #6,R6                           ; R6=UNPACK LOOP COUNTER
006457  115541  013374  146467  ECCCA:  BIC\F   #176000,R1\N,BUF %JZRO  ECCCB   ; IF EQ 0 THEN DONE/ELSE STORE UNPACKED RESIDUE
006460  003740  000004  136466          MOV     #4,Q            %CALL   SHFRES  ; ADJUST HI 4 BITS OF R1
006461  033440  000001  136760          MOV     R1,R0           %CALL   GETRES  ; SAVE IN R0/GET NEXT RESIDUE
006462  115541  003374  136761          BIC     #176000,R1\N,BUF %CALL  SETBAR  ; STORE UNPACKED RESIDUE
006463  003740  000012  126466          MOV     #10.,Q          %CALL   SHFRES  ; ADJUST LO 6 BITS OF PACKED RESIDUE
006464  033141  003000  136760          OR      R0,R1,BUF       %CALL   GETRES  ; FORM RESIDUE, STORE, GET NEXT RESIDUE
006465  031446  000006  116457          DEC     R6              %JMP    ECCCA   ; DECR COUNTER/GO TEST FOR DONE

006466  134541  000374  115501  SHFRES: AND     #176000,R1      %JMP    S.RRR1  ; ISOLATE PACKED PART OF RESIDUE & SHIFT

006467  131742  000010  010000  ECCCB:  neg     #8.,r2                          ;r2=power of alpha in divisor
006470  107740  000374  010000          com     #176000,q                       ;q=1023. for log range check
006471  033741  000036  000000  ecc0a:  MOV     #ELPN+1,R1                      ; ADDRESS OF 2ND ECC RESIDUE
006472  013740  007035  135577          MOV     #ELPN,BAR       %call   s.ldr5  ;high order term in dividend
006473  033740  000017  126744          mov     #15.,r0         %call   poly    ;perform division
006474  010542  007112  000000          add     #Sy,r2\n,bar                    ;Set up pointer to store syndrome
006475  112542  000010  010000          cmp     #8.,r2                          ;test for completion
006476  120342  013005  016471          incb    r5\o,r2,buf      %jneq ecc0a    ;store syndrome
                                                                                ;Set up for next division
                                                                                ;continue if not complete
```



```
                                ;
                                ;*********************************************************************
                                ;;Single Error Test
                                ;
                                ;If there is only one error present then the remainder from
                                ;the Syndrome calculation has only one term.That is,each power series
                                ;contains one term. So that
                                ;           (1)j(0)                 (2)j(0)                 (3)j(0)
                                ;S(1)=Alpha       Y(1), S(2)=Alpha       Y(1),S(3)=Alpha       Y(1)
                                ;                       (i)j(0)
                                ;and in general S(i)=Alpha       Y(1).
                                ;
                                ;It can be seen from this that dividing any successive pairs of syndromes
                                ;             j(0)
                                ;yields Alpha     .This is the antilog of the error location.
                                ;                                3j(0)           2j(0)           j(0)
                                ;For example S(3)/S(2) = Alpha    Y(1)/Alpha    Y(1) = Alpha
                                ;
                                ;Therefore if a Single error is present each division of a Syndrome
                                ;by the sucessive Syndrome will yield the same value and that value will
                                ;be the antilog of the error location.The Single error test then performs
                                ;these sucessive syndrome divisions until either the result of one division
                                ;is not equal to the result of the previous divisions or all the divisions
                                ;produce the same result. If any Syndrome is zero then a single error can
                                ;not exist.
                                ;                                       (0)j(0)
                                ;In the case of a single error,S(0) = Alpha      Y(1) = Y(1)
                                ;Therefore,in the single error case Syndrome number zero is the error value.
                                ;
                                ;*********************************************************************
                                ;
006477  033743  007122  125567  ECC1:   MOV     #SY+8.,R3,BAR   %CALL   S.ldcr  ;Start with SY(8)
006500  033700  010003  116520          MOV     (BUF),R0        %JZRO   ECC2    ;R0=LOG(SY(8))
006501  033743  007121  125567          MOV     #SY+7,R3,BAR    %CALL   s.ldcr  ;Get SY(7) for 1st divide
006502  033707  010003  156520          MOV\F   (BUF),R7        %JZRO   ECC2    ;R7=LG(SY(7))
006503  131140  000007  000000          SUB     R7,R0                           ;DIVIDE SY(8)/SY(7)
006504  030060  030000  156505          ADD\T   R0,Q,R0         %TNEG           ;range test on log sum

                                ;       The Syndromes must form a geometric progression with "distance" R0

006505  033741  000017  116512          MOV     #15.,R1         %JMP    ECC1B   ;Init loop count, enter loop
006506  030067  030007  146507  ECC1A:  ADD\T   R7,Q,R7         %TNEG           ;Normalize descending predictor
006507  031443  007003  125567          DEC     R3,R3,BAR       %CALL   S.LDCR  ;Get next syndrome in CAR
006510  016507  010003  106520          XOR     (BUF),R7\N      %JZRO   ECC2    ;no single error with a zero syndrome
006511  031441  010001  006520          DEC     R1              %JNZRO  ECC2    ;No single error if no geom prog
006512  131147  010000  016506  ECC1B:  SUB     R0,R7           %JNZRO  ECC1A   ;Advance predictor & loop

                                ;       SINGLE ERROR FOUND

006513  013740  007071  125616          MOV     #ELOC+1,BAR     %CALL   S.STR0  ;ERROR LOCATION
006514  013740  007134  125631          MOV     #ECOUNT,BAR     %CALL   INIT1   ;ERROR COUNT = 1
006515  013740  007112  135601          MOV     #SY,BAR         %CALL   S.LDR7  ;FETCH SY(0)
006516  013740  007047  125625          MOV     #EVAL+1,BAR     %CALL   S.STR7  ;ERROR VALUE = SY(0)
006517  013440  000000  106675          NOP                     %JMP    E900    ;go correct error in buffer
```

```
                                ;*********************************************************
                                ;Compute Error Location Polynomial from Syndromes
                                ;
                                ;The following 4 sections of code implement the Berlekamp
                                ;algorithm for generating an Error Location Polynomial.It is
                                ;an iterative algorithm and executes 17 passes. The algorithm
                                ;is documented in 'Peterson and Weldon' and should be understood
                                ;in detail before changing this code.
                                ;
                                ;Basicly it involves generating a polynomial from the syndromes
                                ;whose roots are the locations of the errors.The current iteration
                                ;is n.The error location polynomial at iteration n is elpn. It is
                                ;necessary to keep around the results of one of the past iterations.
                                ;This previous iteration number is m. The previous error location
                                ;polynomial is elpm. A set of scratch values for intermediate calculations
                                ;are stored with the suffix o,such as elpo.
                                ;
                                ;Dism,disn are the 'discrepancy' of elpn and elpm.
                                ;Ln,Lm are measures of the linear independence of the matrix of
                                ; equations which generate elpn and elpm.
                                ;Dn,Dm are the degree of the polynomials elpn and elpm
                                ;
                                ;Each iteration takes elpn,elpm,disn,dism and generates elp(n+1)
                                ;Part 2 takes elpn and the syndromes and computes the
                                ;discrepancy between elpn and elp(n+1).
                                ;Part 3 uses this discrepancy and elpm,dism to generate elp(n+1)
                                ;Part 4 generates l(n+1),d(n+1) and determines what the new
                                ;elpm should be for the next iteration
                                ;
                                ;This code is for the multiple error cases.
                                ;
                                ;*********************************************************
                                ; Part 1 - Set up initial parameters
                                ;
                                ; N is initialized to 0
                                ;                                    2
                                ; Elpn is initialized to 1 + 0x + 0x + .....
                                ;
                                ; M is initialized to -1
                                ;                                    2
                                ; Elpm is initialized to 1 + 0x + 0x + .....
                                ;
                                ; Dism is initialized to 1
                                ;*********************************************************
006520  033741  000011  000000  ecc2:   mov     #9.,R1                          ; [U52EC1]R1=9
006521  033742  000035  000000          mov     #elpn,r2
006522  033746  000001  000000          mov     #1,r6
006523  010542  007011  135624  ecc2a:  add     #elpm-elpn,r2\n,bar %call s.str6
006524  010542  007033  135624          add     #elpo-elpn,r2\n,bar %call s.str6
006525  120342  007002  135624          incb    r2\o,r2,bar     %call   s.str6
006526  031441  000001  000000          dec     R1                              ; [U52EC1]DECR R1
006527  034446  010006  006523          clr     r6              %jnzro  ecc2A
                                ; ***   NOTE - R1 = 0 EXITING LOOP, BECOMES INITIAL VALUE OF N ***
006530  013740  007240  135630          MOV     #LN,BAR         %CALL   S.CLRB  ; [U52EC1]LN=0
```

```
006531  013740  007061  135632          MOV     #M,BAR          %call   initm1  ; [U52EC1]M=-1
006532                                  assume  <dism&17>,eq,<m&17+1>
006532                                  assume  <disn&17>,eq,<dism&17+1>
006532  013740  003001  135630          mov     #1,buf          %call   s.clrb  ; dism = 1 / disn = 0
                        ;               MOV     #DM,BAR         %CALL   S.CLRB  ; [U52EC1]ZAP DM
006533  013740  007064  125630          mov     #LM,bAR         %call   s.clrb  ; LM=0
                        ;               MOV     #DN,BAR         %CALL   S.CLRB  ; [U52EC1]ZAP DN
006534  013740  007134  135630          mov     #ecount,bar     %call   s.clrb  ; ecount = 0

                                ;*****************************************************
                                ;Part 2 - Compute Discrepancy between Sigma(n) to Sigma (n+1)
                                ;
                                ; Disn = elp(0)S(-8+n) + elp(1)S(-8+n+1) + ...+ elp(ln)S(-8+n+ln)
                                ;
                                ;*****************************************************
006535  013740  007240  125576  e311:   mov     #ln,bar         %call   s.ldr3  ;r3=ln(linear independence
                                                                                ;measure of elpn )
006536  033440  000001  000000          MOV     R1,R0                           ; [U52EC1]R0=ITERATION NUMBER
006537  030540  000102  000000          add     #sy0-8.,r0                      ;r0 -> syndrome(-8+n)
006540  033747  000035  126751          mov     #elpn,r7        %call   dot     ;take dot product of elpn,S
006541  033743  010070  106577          mov     #elpo,r3        %jzro   e350    ;if disn=0, skip updating of sigma

                                ;*****************************************************
                                ;Part 3 - Compute Sigma(n+1) from Sigma(n)
                                ;
                                ;For disn not equal to 0:
                                ;
                                ;                    disn  n-m
                                ; Elp(n+1) = elpn + ------ X    elpm
                                ;                    dism
                                ;*****************************************************

006542  013740  007063  125626          mov     #disn,bar       %call   s.st10  ;store discrepancy from above
006543  013740  007061  135573          mov     #m,bar          %call   s.ldr0  ; [U52EC1]ro = m
006544  132140  000001  000000          RSUB    R1,R0                           ; [U52EC1]ro = n-m

                                ;       Note that at this point ELPO = ELPN from the initial conditions or from
                                ;       the end of the last iteration - we now modify ELPO to be ELP(N+1).

006545  030143  000000  010000          add     r0,r3                           ;index into proper position in elpo
006546  013740  007063  125567          mov     #disn,bar       %call   s.ldcr  ;fetch disn
006547  033702  000003  010000          mov     (BUF),r2                        ;r2 = log(disn)
006550  013740  007062  135567          mov     #dism,bar       %call   s.ldcr  ;fetch dism
006551  033705  000003  000000          mov     (BUF),r5                        ;r5 = log(dism)
006552  131142  000005  000000          sub     r5,r2                           ;compute disn/dism
006553  030062  030002  146554          add\t   q,r2            %tneg           ;log range test
006554  033750  000046  010000          mov     #elpm,r10                       ;r10 points to elpm
                        e335:   ;Multiply disn/dism by elpm term
006555  120350  007010  126735          INCB    R10\0,R10,BAR   %CALL   MULBR2  ;SET UP AND PERFORM MULTIPLY
                                ;Xor result of multiply with proper elpn term (result in r5)
                                ;to compute elp(n+1) term
006556  120343  007003  010000          incb    r3\0,r3,bar                     ;fetch proper elp(n+1) term
006557  036505  000003  010000          xor     (buf),r5                        ;perform xor
006560  112543  000101  010000          cmp     #elpo+9.,r3                     ;check for end of loop
```



```
006561  013440  013005  016555          mov     r5,buf          %jnzro  e335    ;Store result

                                ;*************************************************************
                                ;part 4 - Update variables for next round of elpn computation.
                                ;         Now that elp(n+1) has been computed from elp(n)
                                ;         it is necessary to compute ln(n+1),dn(n+1),dm(n+1),elpm(n+1),
                                ;         and dism(n+1) from ln,dn,dm,elpm,and dism.
                                ;*************************************************************

                                ;On entry r0 contains n-m
                                ;Compute Dn(n+1) = dm + n - m

                                ;       mov     #dm,bar         %call   s.ldr2  ; [U52EC1]get dm in r2
                                ;       add     r0,r2                           ; [U52EC1]R2 = dm + n - m = dn(n+1)

                                ;Now see if Ln will change - i.e. if Lm+n-m > Ln

006562  013740  007064  135576          mov     #lm,bar         %call   s.ldr3  ;R3 = Lm
006563  030140  000003  010000          add     r3,r0                           ;R3 = Lm+n-m = new L(n+1) candidate
006564  013740  007240  135577          mov     #ln,bar         %call   s.ldr5  ;R5 = Ln
006565  011140  000005  010000          subc    r5,r0\n                         ;Compare to see if Ln changes
006566  013440  033000  156575          mov\f   r0,buf          %jneg   e345    ;br if not, else do it

                                ;       ELPM should be updated if n-Ln > m-Lm, but this is equivalent to
                                ;       Lm+n-m>Ln so the test has been made already! Go for it!

006567  033747  000035  000000          mov     #elpn,r7
006570  033746  000046  136670          mov     #elpm,r6        %call   ecmov9  ;elpm(n+1)=elp(n)
006571  013740  007063  135600          mov     #disn,bar       %call   s.ldr6  ;dism(n+1)=disn
006572  013740  007062  125624          mov     #dism,bar       %call   s.str6  ;store new dism
006573  013740  007064  135623          mov     #lm,bar         %call   s.str5  ;lm(n+1)=ln
006574  013740  007061  135620          MOV     #M,BAR          %CALL   S.STR1  ; [U52EC1]M(N+1)=N
                        ;               mov     #n,bar          %call   s.ldr0  ; [U52EC1]R0 = n
                        ;               mov     #m,bar          %call   s.str0  ; [U52EC1]m(n+1) = n
                        ;               mov     #dn,bar         %call   s.ldr3  ; [U52EC1]dm(n+1)=dn
                        ;               mov     #dm,bar         %call   s.str3  ; [U52EC1]store new dm
                                ;
                                ;Now that elpn has been used for all needed computations
                                ;replace elpn with the computed elp(n+1)
                                ;
006575  033746  000035  010000  e345:   mov     #elpn,r6
006576  033747  000070  126670          mov     #elpo,r7        %call   ecmov9  ;Store elp(n+1)
                        ;               Mov     #dn,bar         %call   s.str2  ; [U52EC1]Store new Dn
006577  112541  000020  010000  e350:   CMP     #16.,R1                         ; [U52EC1]CHECK FOR END OF MAIN LOOP
006600  130441  010001  016535          INC     R1              %jnzro  e311    ; [U52EC1]loop


                                ;Solve Error Location Polynomial to find locations and values
                                ;
                                ;Part 1 - Factor Error Location Polynomial

                                ;*****************************************************;
```

```
                                        ;The Error Location Polynomial(elp) is of the form :
                                        ;                              2                 v
                                        ;   elp = 1 + elp(1)x + elp(2)x + ... + elp(v)x = 0
                                        ;
                                        ;   where v is the number of errors.
                                        ;
                                        ;   This is also of the form:
                                        ;            i(1)          i(2)             i(v)
                                        ;   (x + alpha    )(x+alpha    )...(x+alpha    ) = 0
                                        ;
                                        ;   where i(.) are the error locations ( in symbol number )
                                        ;
                                        ;  Values are plugged into the formula until a solution is found.
                                        ;
                                        ;The Error Location Polynomial Prime (elpp) is the elp
                                        ;after one of the error locations has been factored out. It has the
                                        ;form:                               2                    v-1
                                        ;      elpp = 1 + elpp(1)x + elpp(2)x + ... + elpp(v-1)x
                                        ;
                                        ;                     i(1)
                                        ;  and elp = (1+alpha    x)(elpp)
                                        ;
                                        ; Representing the coefficients of elp in terms of the coefficients
                                        ; of elpp allows us to find the elpp coefficients:
                                        ;              i(1)                      2                   v-1
                                        ; elp = (1+alpha    x)(1 + elpp(1)x + elpp(2)x +...+ elpp(v-1)x
                                        ;                     i(1)                       i(1)          2
                                        ; or elp = 1 + (alpha    +elpp(1))x + (elpp(1)alpha    +elpp(2))x
                                        ;
                                        ;                         i(1)            3                      i(1) v-1
                                        ;          + (elpp(2)alpha    + elpp(3))x +...+ (elpp(v-1)alpha    )x
                                        ;Since this is a representation of elp the coefficients of each term
                                        ;in x must be the same the coefficients in the original representation
                                        ;of elp.Equating the coefficients yields:
                                        ;                                i(1)
                                        ; elpp(1) = elp(1) + elpp(0)alpha
                                        ;                                  i(1)
                                        ; elpp(i) = elp(i) + elpp(i-1) alpha
                                        ;                                  i(1)
                                        ; elpp(v) = elp(v) + elpp(v-1) alpha
                                        ;                                           i(1)
                                        ;Note however that elpp(v) must be zero when alpha    is a root
                                        ;Therefore, by going through an iterative procedure rather than a
                                        ;straight substitution it can be determined what the root is and
                                        ;simultaneously find the coefficients of elpp.
                                        ;
                                        ;The algorithm then is as follows:
                                        ;                (0)
                                        ;Start with alpha
                                        ;Using the outined iterative procedure find elpp(1) through elpp(v)
                                        ;If elpp(v) is zero then the alpha value picked is a root and the
                                        ;values generated for elpp are valid. If not increment the power
                                        ;of alpha and redo the iteration. Continue until a root is found
                                        ;The section of code after this one is then executed ( after each root
```



```
                                        ;is found) in order to determine the error value corresponding to that
                                        ;error location. The elpp values then become the elp values and the
                                        ;procedure is repeated. This loop continues until all the errors are found.

                                        ;******************************************************************

006601  013740  007240  125575                  MOV     #LN,BAR         %CALL   S.LDR2  ;r2 = v
006602  131542  010011  116734                  SUB     #9.,R2          %JZRO   ERRRR   ;IF EQ 0 THEN FATAL/ELSE test for uncorecctable erro
006603  034442  020002  156734                  clr\F   r2              %JGTE   ERRRR   ;branch if uncorrectable
                                        ;r2 will be power of alpha tested as root
                                        ;valid range for power is 0 to 1023.
006604  013740  007240  125601          A:      MOV     #LN,BAR         %CALL   S.LDR7  ; R7 = CURRENT VALUE OF V
006605  112042  000002  010000                  CMP     R2,Q                            ;OUTSIDE VALID RANGE-ERROR
006606  113542  027070  116734                  BIS     #ALGADR,R2\N,BAR %JGTE  ERRRR   ; TAKE ADV OF FACT THAT ELPP(0)=1
006607  033750  000060  125577                  MOV     #ELPP+1,R10     %CALL   S.LDR5  ; R10=ELPP INDEX,R5=ALOG(R2)
006610  033740  007036  116614                  MOV     #ELPN+1,R0,BAR  %JMP    BP      ;r0 = elpn index, enter loop in middle
                                        B:      ;Multiply elpp(.)*alpha**(i(.))
006611  013445  000005  000000                  TST     R5
006612  113545  017074  036737                  BIS     #LGADR,R5\N,BAR %CNZRO  MULR2   ; PERFORM MULTIPLY
006613  033440  007000  000000                  MOV     R0,R0,BAR
006614  036505  000003  010000          BP:     XOR     (BUF),R5                        ;xOR RESULT WITH ELPN TERM
006615  120350  007010  010000                  INCB    R10\0,R10,BAR                   ;SETUP NEXT ELPP TERM

                                        ;Continue until elpp(v) has been computed

006616  021347  003005  010000                  DECB    R5\0,R7,BUF                     ;STORE ELPP, DECREMENT LOOP COUNTER
006617  130460  010000  056611                  INC\T   R0,R0           %JNZRO  B       ;CONTINUE
006620  013445  000005  000000                  TST     R5                              ;DONE - CHECK REMAINDER FOR ZERO
006621  130462  010002  046604                  INC\T   R2              %JNZRO  A       ;TEST ELPP(V) FOR 0, LOOP IF NOT
                                                                                        ;IF ZERO,ROOT HAS BEEN FOUND
                                                                                        ;AND JUMP TO FIND ERROR VALUE
                                        ;If power of alpha chosen was not a root,increment power
                                        ;of alpha and try again


                                        ;Part 2 - Use error location to find error value
                                        ;*************************************************************
                                        ;This code implements the following algorithm:
                                        ;
                                        ;
                                        ;        S(0)elpp(v-1) + S(1)elpp(v-2) + ... + S(v-1)elpp(0)
                                        ;value = -------------------------------------------------------
                                        ;             (0)                i(1)                (v-1)i(1)
                                        ;        Alpha    elpp(v-1) + Alpha    elpp(v-2) +...+ Alpha        elpp(0)
                                        ;
                                        ;*************************************************************

006622  013740  007134  125602                  MOV     #ECOUNT,BAR     %CALL   S.LD10  ;INCREMENT NUMBER OF ERRORS
006623  130450  000010  135626                  INC     R10             %CALL   S.ST10  ; AND RESET
006624  030550  007070  135621                  ADD     #ELOC,R10,BAR   %CALL   S.STR2  ;STORE ERROR LOCATION
006625  033741  007057  135631                  MOV     #ELPP,R1,BAR    %CALL   INIT1   ;ELPP(0) = 1
006626  013740  007240  135614                  MOV     #LN,BAR         %CALL   S.DECB  ; R0=V-1
006627  033443  000000  010000                  MOV     R0,R3                           ; R3=V-1
                                        ;
```


LSCS FORM=QUAD

```
                                ;computation of denominator
                                ;This is really the evaluation of the polynomial
                                ;        (v-1)                (v-2)
                                ;elpp(0)X      + elpp(1)X     +...+ elpp(v-1) = 0
                                ;              i(1)
                                ;at X = alpha     .
                                ;
006630  034445  000005  136744          CLR     R5              %CALL   POLY    ;RETURNS R5=DENOMINATOR
006631  033441  000005  000000          MOV     R5,R1                           ;SAVE R5 BEFORE NEXT CALL
                                ;
                                ;Now compute numerator
                                ;This is the dot product of a vector composed of
                                ;elpp(0),elpp(1),elpp(2),....,elpp(v-1) with the vector
                                ;S(v-1),S(v-2),S(v-3),...,S(0)
                                ;
006632  033747  010057  116734          MOV     #ELPP,R7        %JZRO   ERRRR   ;R7 POINTS TO ELPP VECTOR
006633  033740  000112  000000          MOV     #SY0,R0                         ;R0 IS POINTER TO SYNDROMES
006634  030140  000003  136751          ADD     R3,R0           %CALL   DOT     ;R0 IS PTR TO SYNDROME VECTOR
006635  013440  010000  106734          NOP                     %JZRO   ERRRR
006636  113550  007074  135601          BIS     #LGADR,R10\N,BAR %CALL  S.LDR7  ; [U52EC1]DOT RETURNS R10 = NUMERATOR/R7 = LOG (NEMERATOR)
                                ;;;;    BIS     #LGADR,R1\N,BAR %CALL   S.LDR5
                                ;
                                ;  Compute value = numerator/denominator
                                ;
006637  013450  000010  000000          TST     R10
006640  113541  017074  106734          BIS     #LGADR,R1\N,BAR %JZRO   ERRRR   ; IF EQ 0 THEN ERROR/ELSE, LOAD CAR WITH DENOMINATOR
006641  013441  000001  000000          TST     R1
006642  033705  010003  106734          MOV     (BUF),R5        %JZRO   ERRRR   ; IF EQ 0 THEN ERROR/ELSE r5 = log(denominator)
006643  132145  000007  136741          RSUB    R7,R5           %CALL   MULR2X  ; DIVIDE
                                ;
                                ;    Returns r5 = error value
                                ;
006644  013740  007134  125602          MOV     #ECOUNT,BAR     %CALL   S.LD10  ;COMPUTE STORAGE LOCATION
006645  010550  007046  135623          ADD     #EVAL,R10\N,BAR %CALL   S.STR5  ; [U52EC1]store error value


                                ;Part 3 - Modify Syndromes and return to find next factor(location)

                                ;***************************************************************
                                ;
                                ;Each syndrome is really the summation of a power series
                                ;comprised of the error locations and values. For instance
                                ;for v errors,                          2       2       2
                                ;               Syndrome(2) = X(1) Y(1) + X(2) Y(2) +...+X(v) Y(v)
                                ;At this point in the code, an error location and value have
                                ;been found and factored out of the elp.Since the syndromes and
                                ;elp coefficients are related,and the elp coefficients have been
                                ;updated to represent v-1 errors, the syndromes must now be updated
                                ;to represent v-1 errors.The situation now is as if there were only
                                ;v-1 errors to begin with and the syndromes for the case of v-1
                                ;errors must be derived from the syndromes for the v error case.
                                ;From the form of the syndromes it is clear that the form of the
                                ;new syndromes will be                  2       2       2
                                ;                       Syndrome(2) = X(2) Y(2) +X(3) Y(3)+...+X(v) Y(v)
```



```
                                ;                                                 2
                                ;Therefore the transformation is S'(2) = S(2) - X(1) Y(1)
                                ;                                                 i
                                ;Or in general    S'(i) = S(i) - X(1) Y(1)
                                ;*****************************************************************

                                ;r5 is error value on entry to this module

006646  033740  000112  000000          MOV     #SY0,R0                         ;R0 IS POINTER TO SYNDROME
                                ;only need to udate those syndromes which will be needed next time
006647  013740  007240  125576          MOV     #LN,BAR         %CALL   S.LDR3  ;R3=NUMBER OF SYNDROMES TO FIX
006650  010550  007070  135575          ADD     #ELOC,R10\N,BAR %CALL   S.LDR2  ; [U52EC1]R2=POWER OF ALPHA(ROOT)
006651  133545  007074  135601          BIS     #LGADR,R5,BAR   %CALL   S.LDR7  ; R5 IS ERROR VALUE/R7=LOG(ERROR VALUE)
006652  034446  000006  000000          CLR     R6                              ;R6 IS I FROM DISCUSSION
006653  033445  000006  010000  F:      MOV     R6,R5                           ;SET UP FOR MULTIPLY
006654  030145  000007  136741          ADD     R7,R5           %CALL   MULR2X  ;multiply X(.)**i(Y(.))
                                ;returns result in r5
006655  030146  000002  000000          ADD     R2,R6                           ;SET UP NEXT POWER OF x(.)
006656  132046  000006  000000          RSUB    R6,Q,R6                         ;RANGE TEST
006657  030066  030006  156660          ADD\T   Q,R6            %TNEG           ;BRING LOG INTO RANGE
006660  120340  007000  010000          INCB    R0\0,R0,BAR                     ;bar -> SYNDROME
006661  036505  000003  010000          XOR     (BUF),R5                        ;FORM MODIFIED SYNDROME IN r5
006662  031443  000003  000000          DEC     R3                              ;COUNT DOWN THE LOOP
006663  013440  023005  106653          MOV     R5,BUF          %JGTE   F       ;STORE SYNDROME,CONTINUE
                                ;elpp parameters computed above become the elpn
                                ;parameters for the next pass

006664  013740  007240  125576          MOV     #LN,BAR         %CALL   S.LDR3  ;NUMBER OF TERMS TO TRANSFER
006665  033746  010035  106675          MOV     #ELPN,R6        %JEQ E900       ;SET UP ELPN POINTER
                                                                                ;TEST FOR NO MORE ERRORS
006666  033747  000057  136672          MOV     #ELPP,R7        %CALL   ECMOVE  ;TRANSFER ELPP TO ELPN
006667  013440  000000  106604          NOP                     %JMP    A       ;FIND NEXT ERROR

                                ;*****************************************************************
                                ;
                                ;Ecmove subroutine
                                ;
                                ;This subroutine moves elements of one list into another list
                                ;inputs:        r7 points to input list
                                ;               r6 points to output list
                                ;ecmov9 moves 9 elements from one list to the other list
                                ;ecmove moves a number of elements specified in r3
                                ;
006670  033743  000010  116672  ECMOV9: MOV     #8.,R3          %JMP    ECMOVE
006671  013440  033010  137777  ECMOVX: MOV     R10,BUF         %RNEG
006672  120347  007007  125602  ECMOVE: INCB    R7\0,R7,BAR     %CALL   S.LD10
006673  120346  007006  010000          INCB    R6\0,R6,BAR
006674  031443  000003  116671          DEC     R3              %JMP    ECMOVX


                                ;
                                ;*****************************************************
                                ;Amend Data Buffer
                                ;This section steps through the lists of error locations and
```

```
                                ;values and corrects the data buffer.
                                ;*****************************************************

006675  013740  007172  135573  E900:   MOV     #SAVR7,BAR      %CALL   S.LDRO  ; RO=PTR TO BUFFER CONTROL AREA
006676  030540  007002  135573          ADD     #BUF.BP,RO,BAR  %CALL   S.LDRO  ; RO=START OF DATA BUFFER
006677  033741  000071  000000          MOV     #ELOC+1,R1                      ;START OF ERROR LOCATION LIST
006700  033742  000047  000000          MOV     #EVAL+1,R2                      ;START OF ERROR VALUE LIST
006701  013740  007134  125576          MOV     #ECOUNT,BAR     %CALL   S.LDR3  ;NUMBER OF ERRORS
006702  120341  007001  125601  E910:   INCB    R1\0,R1,BAR     %CALL   S.LDR7  ;NEXT ERROR LOCATION
006703  131547  000377  010000          SUB     #255.,R7
                                ; ***   VALUE USED IN NEXT INST = <<<WORDS/SECTOR*16.+9.>/10.+17.>> ***
006704  131547  000256  000000          SUB     #<429.-255.>,R7
006705  077450  030007  016734          COM\L   R7,R10          %JNNEG  ERRRR   ; IF NOT NEGATIVE THEN ERROR
006706  030150  000010  010000          ADD     R10,R10                         ; MULTIPLY ERROR LOCATION BY 10
006707  001150  000007  000000          SUBC    R7,R10,Q                        ; & SEPARATE INTO WORD AND BIT LOCATIONS
006710  074650  000007  000000          AND\L   #7,Q,R10                        ; SET R10
006711  075647  000007  010000          BIC\L   #7,Q,R7                         ; SET R7
006712  003740  000004  125503          MOV     #4,Q            %CALL   S.RRR7  ;BUFFER OFFSET
006713  120342  007002  135600          INCB    R2\0,R2,BAR     %CALL   S.LDR6  ;ERROR VALUE
006714  007740  000000  116716          COM     #0,Q            %JMP    E917    ;q = -1/CONTINUE
006715  063446  030006  146717  E915:   MOV\LQF R6,R6           %JNEG   e920    ; R6 ROTATES LEFT, Q DOUBLES
006716  031450  000010  116715  E917:   DEC     R10             %JMP    E915    ;SHIFT ERROR VALUE FOR MASK
006717  034045  000006  010000  E920:   AND     R6,Q,R5                         ;LOW ORDER MASK
006720  007240  000000  000000          COM     Q
006721  034046  000006  010000          AND     R6,Q,R6                         ;HIGH ORDER MASK
006722  030147  007000  135572          ADD     R0,R7,BAR       %CALL   S.LDQQ  ;BUFFER ADDRESS TO MASK
006723  006045  000005  135613          XOR     R5,Q            %CALL   S.STQQ  ; [U52EC2]CORRECT DATA
006724  130447  007007  135572          INC     R7,R7,BAR       %CALL   S.LDQQ
006725  006046  000006  135613          XOR     R6,Q            %CALL   S.STQQ  ; [U52EC2]CORRECT DATA
006726  031443  000003  000000          DEC     R3
006727  034443  010003  046702          CLR\F   R3              %JNZRO  E910

006730  013740  007172  135601  ERET:   MOV     #SAVR7,BAR      %CALL   S.LDR7  ; RESTORE BUFFER POINTER TO R7
006731  010547  007006  135577          ADD     #BUF.SD,R7\N,BAR %call  S.LDUB  ; RESTORE SDI CTL BLOCK PTR TO UBAR
006732  010545  007022  135602          ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; RESTORE MSCP PKT PTR
006733  013443  000003  127777          TST     R3              %RET            ; R3=0(OK) OR 1(ERROR)/RETURN


006734  033743  000001  106730  ERRRR:  MOV     #1,R3           %JMP    ERET    ; R3=1/GO RESTORE R7/RETURN



                                ;subroutines
                                ;MULR2 : this routine multiplies r5 by alg(r2)
                                ;        input: lgadr(r5) in BAR and state reflected in condition codes
                                ;               r2 is log of other multiplicand
                                ;               q contains 1023.
                                ;        output: result in R5
                                ;MULBR2: input: BAR points to place to get R5

006735  033705  000003  000000  MULBR2: MOV     (BUF),R5
006736  113545  017074  127777          BIS     #LGADR,R5\N,BAR %RZRO           ;TEST FOR 0 MULTIPLICAND/ELSE, GET MULTIPLICAND FROM MEMORY
006737  033705  000003  050000  MULR2:  MOV\F   (BUF),R5                        ;TEST FOR 0 MULTIPLICAND
                                                                                ;TAKE LOG IF NOT ZERO
```



```
006740  030145  000002  000000          ADD     R2,R5
006741  132045  030005  156742  MULR2X: RSUB\F  R5,Q,R5         %TNEG           ;RANGE TEST ON LOG SUM
006742  030065  030005  156743          ADD\T   Q,R5            %TNEG           ;BRING LOG INTO RANGE
006743  133545  007070  105577  MUL2RA: BIS     #ALGADR,R5,BAR  %JMP    S.LDR5  ;TAKE ANTILOG AND RETURN

                                ;POLY : this routine evaluates a polynomial
                                ;
                                ;Note that evaluating a polynomial at say x = r is the same
                                ;as the remainder after dividing the polynomial by x-r.
                                ;r2 = value at which polynomial is being evaluated
                                ;r5 = polynomial result up to that point in calculation
                                ;r1 = term pointer in polynomial
                                ;r0 = number of terms - 1
                                ;
                                ;outputs reslut in r5
                                ;
006744  013445  000005  116746  POLY:   TST     R5              %JMP    POLYX
006745  036505  030003  137777  POLYP:  XOR     (BUF),R5        %RNEG
006746  113545  017074  036737  POLYX:  BIS     #LGADR,R5\N,BAR %CNZRO  MULR2
006747  120341  007001  010000          INCB    R1\0,R1,BAR
006750  031440  000000  116745          DEC     R0              %JMP    POLYP


                                ;DOT : this routine takes the dot product of two polynomials

006751  034450  000010  106753  DOT:    CLR     R10             %JMP    DOT2
006752  036150  030005  127777  DOT1:   XOR     R5,R10          %RNEG
006753  120347  007007  135577  DOT2:   INCB    R7\0,R7,BAR     %CALL   S.LDR5
006754  113545  007074  125575          BIS     #LGADR,R5\N,BAR %CALL   S.LDR2
006755  013445  000005  000000          TST     R5
006756  021340  017000  026735          DECB    R0\0,R0,BAR     %CNZRO  MULBR2
006757  031443  000003  116752          DEC     R3              %JMP    DOT1

                                ; ***   ROUTINES FOR UNPACKING ECC RESIDUES ***
006760  021343  007003  135574  GETRES: DECB    R3\0,R3,BAR     %CALL   S.LDR1  ; R1=PACKED ECC RESIDUE/UPDATE POINTER
006761  021342  007002  137777  SETBAR: DECB    R2\0,R2,BAR     %RET            ; BAR=PTR TO RESIDUE STORAGE AREA, UPDATE R2/RETURN
                                        .PAGE
```

```
                                .SBTTL  U.PROC REVECTORING PROCESS
                            ;;+     19-SEPT-1983    UDA/QDA MICROCODE
                            ;
                            ; REVECTORING CODE
                            ;
                            ; THIS CODE IS STRUCTURED AS A PROCESS WHICH RUNS SOMETIMES UNDER THE UNIBUS
                            ; PROCESSOR AND SOMETIMES UNDER THE DISK PROCESSOR.  IT IS DISPATCHED TO FROM
                            ; THE IDLE LOOPS OF THESE PROCESSORS.
                            ;
                            ; WHEN REVECTORING IS DESIRED FOR A DRIVE, THE "RVCT" BIT IS SET IN ITS SDI
                            ; CONTROL BLOCK STATUS WORD (SDI.ST). WHEN THE REVECTORING CONTEXT BLOCK
                            ; (RAM LOCATIONS REVSTR THROUGH REVEND) BECOMES FREE (REVSDI=0) THEN REVSDI
                            ; IS SET TO POINT TO THE SDI NEEDING REVECTORING AND THIS PROCESS IS ENTERED
                            ; AT U.RVC1 BY THE UPPER PROCESSOR. THE VECTORING IS UNDER CONTROL OF THE
                            ; WORD RVCVEC, WHICH ACTS AS A "PC" POINTING TO THE NEXT SECTION TO BE EXECUTED;
                            ; IF RVCVEC IS EVEN A DISK PROCESSOR CODE SECTION IS INVOKED, WHILE IF IT IS
                            ; ODD A UNIBUS PROCESSOR SECTION IS INVOKED; IN EITHER CASE RVCVEC IS A POINTER
                            ; TO A DISPATCH IN THE UDA VECTOR TABLE.
                            ;
                            ; INPUTS:
                            ;               RVCBUF POINTS TO BUFFER NEEDING REVECTORING
                            ;               UBAR/DBAR POINTS TO SDI CONTROL BLOCK
                            ;
                            ; OUTPUTS:
                            ;               SDI.DB ADVANCED PAST BUFFER (IF APPROPRIATE),
                            ;               OPERATION COMPLETE, DISK
                            ;               ON CORRECT CYL/GROUP/TRACK FOR SUBSEQUENT OPERATIONS.
                            ;;-
                                    .PAGE
```



```
                            ;       SINCE THE DISK PROCESSOR INITIALIZES THE VECTOR, THE FIRST STEP
                            ;       IS TO MAKE SURE THE UNIBUS PROCESSOR KNOWS WE ARE IN REVECTORING
                            ;       SO WE EXECUTE A NOP IN THE UPPER PROCESSOR...

                            ;       R1 CONTAINS INCREMENTED VECTOR ADDRESS

                           U.RVC1: ; MOV    R1,BUF          %JMP    U.RVUX  ; [UQA]ACTUALLY DONE IN THE DISPATCH VECTOR...

                                    .PAGE
```

```
                                ;       NOW SWITCH TO THE UPPER PROCESSOR TO SET UP TO READ A BLOCK OF RCT
                                ;       THIS STEP IS INVOKED AT ITS MAIN ENTRY TO START A READ, AND IS ALSO INVOKED
                                ;       AT A SECONDARY ENTRY TO READ SUCCESSIVE COPIES OF THE RCT BLOCK IN CASE
                                ;       AN UNCORRECTABLE READ ERROR OCCURS.  THIS STEP IS PERFORMED IN THE UPPER
                                ;       PROCESSOR BECAUSE THE LBN-->CYL/GRP/TRK CONVERSION ROUTINES ARE HERE
                                ;*NOTE* THIS SECTION IS ENTERED FROM MANY OTHER SPOTS IN THE REVECTORING CODE - BEWARE!

006762  010545  007034  135574  U.RVC3: ADD     #SDI.RC,UBAR\N,BAR %CALL S.LDR1 ; GET WORD CONTAINING # COPIES OF RCT
006763  134541  000017  125504          AND     #<LONIB*256.>,R1 %CALL  S.SWB1  ; ISOLATE # COPIES/SWAP BYTES
006764  013740  007220  125620          MOV     #RVCRTY,BAR     %CALL   S.STR1  ; STORE IT AWAY AS THE RETRY COUNT
006765                                  ASSUME  SDI.L2,EQ,SDI.L1+1              ; MAKE SURE SEQUENTIAL
006765  000545  007076  133211          ADD     #SDI.L2,UBAR,Q,BAR %CALL LDR2R1 ; R2,R1=1ST LBN OF RCT
006766  013740  007224  125600          MOV     #RCTBLK,BAR     %CALL   S.LDR6  ; GET OFFSET TO DESIRED RCT BLOCK IN R6
006767  030546  000002  010000          ADD     #2,R6                           ; [U52EC2][ECO#1]BIAS TO GET PAST RCT HDR BLKS
006770  013740  007226  125601          MOV     #RVCBUF,BAR     %CALL   S.LDR7  ; GET POINTER TO REVECTOR BUFFER IN R7

                                ; ***   ENTER HERE WITH CURRENT LBN IN (R2,R1), RCT SIZE IN R6, R7 -> BUFFER, TO TRY NEXT COPY ***
006771  030141  000006  000000  U.RV3A: ADD     R6,R1                           ; ADD R6 TO (R2,R1)
006772  130462  020002  146773          INC\T   R2,R2           %TCRY

                                ; ***   ENTER HERE WITH LBN TO OPERATE ON IN (R2,R1); CHAR, UBAR-->SDI CTL BLK
006773                                  ASSUME  LBNMSK,EQ,LOBYT                 ; MAKE SURE MASK IS LO BYTE
006773  010545  007074  135604          ADD     #SDI.12,UBAR\N,BAR %CALL S.LLBO ;[V05] GET LBN'S/TRACK IN R0<7:0>

                                ; ***   ENTER HERE WITH BN IN (R2,R1), R0=BLOCKS/TRACK,
                                ; ***   R7 -> BUFFER, UBAR --> SDI CTL BLK
006774  010545  007022  135602  U.RV3C: ADD     #SDI.PQ,UBAR\N,BAR %CALL S.LD10 ; GET MSCP PACKET POINTER INTO R10
006775  010547  007003  125620          ADD     #BUF.HL,R7\N,BAR %CALL  S.STR1  ; SAVE LO BLK NUMBER
006776  010547  007004  125621          ADD     #BUF.HH,R7\N,BAR %CALL  S.STR2  ; SAVE HI BLK NUMBER
006777  135542  000360  132761          BIC     #HDCOD,R2       %CALL   U.IOPX  ; ELIMINATE HEADER TYPE AND CONVERT IT
007000  013740  007226  125601          MOV     #RVCBUF,BAR     %CALL   S.LDR7  ; RESTORE BUFFER PTR TO R7
007001                                  ASSUME  S.CYLH,EQ,S.CYLL+1              ; MAKE SURE SEQUENTIAL
007001  000550  007025  123211          ADD     #S.CYLH,R10,Q,BAR %CALL LDR2R1  ; R2,R1=CYLINDER NUMBER
007002  010547  007005  135572          ADD     #BUF.TA,R7\N,BAR %CALL  S.LDQQ  ; GET SDI LEVEL 1 CMD + HEAD NUMBER
007003  104640  000377  000000          AND     #HIBYT,Q                        ; ISOLATE LEVEL 1 COMMAND FOR U.IOPE
007004  010545  007011  133073          ADD     #SDI.CL,UBAR\N,BAR %CALL U.IOPE ; MOVE CYLINDER TO SDI.CL+SDI.CH, SET SEEK
007005  010547  007005  125613          ADD     #BUF.TA,R7\N,BAR %CALL  S.STQQ  ; SAVE SDI LEVEL 1 CMD + HEAD # IN BUFFER
                                ; ***   IF WRITE COMMAND THEN REFILL BUFFER ***
007006                                  ASSUME  WRITCD,LT,0                     ; MAKE SURE WRITE CODE IS NEG
007006  013440  030000  007016          NOP                     %JNNEG  U.RV3D  ; IF READ THEN CONTINUE
007007  010545  007022  134123          ADD     #SDI.PQ,UBAR\N,BAR %CALL CKWRER ; SET UP OPCODE FOR U.BFLC
007010  010547  007002  135576          ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR3  ; [16K]R3=BUFFER PTR
007011  013440  010000  107016          NOP                     %JZRO   U.RV3D  ; [16K]IF NEQ 0 THEN JUMP TO REFILL BUFFER
007012  010545  007026  135574          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDR1 ; [16K]R1=EXTENDED STATUS
007013  033541  003001  132014          BIS     #RVWRIT,R1,BUF  %CALL   U.VAXP  ; [16K]SET REVECT WRITE FLAG/CHK FOR VAX KLUDGE
007014  013440  000000  114205          NOP                     %JMP    U.BFLC  ; [16K]GO REFILL BUFFER
007015  013440  000000  132014  U.RV3B: NOP                     %CALL   U.VAXP  ; [U52EC2]CHK FOR VAX CLUDGE AGAIN
007016  013740  007250  125574  U.RV3D: MOV     #RVECUP,BAR     %CALL   S.LDR1  ; [UQA]GET PROCESS VECTOR
007017  130441  000001  135620          INC     R1              %CALL   S.STR1  ; [UQA]INCREMENT IT & RETURN TO UPROC IDLE LOOP
007020  013740  007254  135574  U.RVUX: MOV     #RVCFLG,BAR     %CALL   S.LDR1  ; [UQA]GET UPPER/DRIVE FLAG
007021  017440  003001  127777          COM     R1,BUF          %RET            ; [UQA]SWITCH TO D.PROC/RETURN
                                        .PAGE
```



```
                                ;       AT THIS POINT, WE HAVE (WITH LUCK) READ A BLOCK INTO RVBUF
                                ; [16K]BUF.ST IS SET
                                ;       AS FOLLOWS:
                                ;                       BFULL+BLAST             SECTOR HAD NO ERRORS
                                ;                       BFULL+BLAST+BECC        SECTOR HAD AN ECC ERROR
                                ;                       BFULL+BLAST+BECC+BGRUP  SECTOR COULD NOT BE READ
                                ;       THIS STEP INITIATES CORRECTION AND/OR RETRY ON ERRORS, AND PERFORMS
                                ;       THE RCT SCAN IF THE SECTOR DATA IS GOOD.
                                ; NOTE!!!!
                                ;       THIS CODE IS ALSO USED TO CLEAN UP AFTER THE FINAL READ/WRITE OF THE REVECTOR TARGET

                                U.RVC5: ; NOP           @RUPF                   ; [U52EC2]ACTUALLY DONE IN THE DISPATCH VECTOR...

100000                                  WRPFLG  =       BIT15                   ; [U52EC2] RCTOFF FLAG, SET IF WE WRAP AROUND RCT

007022  013740  007226  125601  U.RV57: MOV     #RVCBUF,BAR     %CALL   S.LDR7  ; GET BUFFER POINTER IN R7
007023  010547  007001  135575          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR2  ; [16K]GET BUFFER STATUS IN R2
007024  114542  000020  010000          BIT     #BECC,R2                        ; CHECK FOR ECC OR OTHER ERRORS
007025  014542  010001  117033          BIT     #BGRUP,R2       %JZRO   U.RV5A  ; BRANCH ON NO ERRORS, CHECK ERROR TYPE
007026  014542  010004  107031          BIT     #BNXCOP,R2      %JZRO   U.RV5J  ; [U52EC2]IF BECC ONLY THEN DO CORRECTION
007027  035542  013004  117140          BIC     #BNXCOP,R2,BUF  %JZRO   U.RVFH  ; [U52EC2]IF BNXCOP EQ 0 THEN FATAL ERROR
007030  013440  000000  107047          NOP                     %JMP    U.RV5W  ; [U52EC2]  ELSE GO TO NEXT COPY
                                ; ***   RCT OR RBN BLOCK NUMBER AND ECC ERROR - ONLY CORRECT IF RCT!! ***
007031  034442  060002  076450  U.RV5J: CLR\F   R2              %CNUPF  ECCC    ; [U52EC2]IF NOT RBN THEN DO ECC CORRECTION
                                                                                ; [U52EC2]  ELSE FAKE SUCCESS SO
                                                                                ; [U52EC2]WILL CORRECT AND RETRY IF NECESSARY
007032  033760  010350  057041          MOV\T   #<ST.DAT!SC.ECC>,R0 %JNZRO U.RV5X; IF NOT CORRECTED THEN GO DO RECOVERY
                                ; ***   RCT BLK - NO ECC ERRORS/ECC CORRECTABLE OR RBN - FAKED ECC CORRECTION - CALCULATE AND CHECK EDC ***
007033  010547  007002  135575  U.RV5A: ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR2  ; R2=BUFFER POINTER
007034  033743  060105  117061          MOV     #EDSEED,R3      %JUPF   U.RV5Z  ; [U52EC2]IF RBN I/O THEN SKIP
007035  013440  007002  000000          MOV     R2,BAR                          ; [U52EC2]BAR=DATA BUFFER PTR
007036  133740  000001  135220          MOV     #SECSZ,R0       %CALL   U.CKEB  ; R0=WORD COUNT/GO CALC EDC IF ECC CORRECTED
007037  016503  000003  000000          XOR     (BUF),R3\N                      ; IF EDC'S EQ
007040  033740  010112  157061          MOV\F   #<ST.CNT!SC.EDC>,R0 %JZRO U.RV5Z ; [U52EC2] THEN CONTINUE/ ELSE EDC ERROR
                                ; ***   RCT BLOCK ECC OR EDC ERROR - CHECK IF RECOVERY EXHAUSTED ***
007041  013740  007220  135574  U.RV5X: MOV     #RVCRTY,BAR     %CALL   S.LDR1  ; R1=RETRY/RECOVERY WORD
007042  114541  000004  010000          BIT     #BIT10,R1                       ; IF RETRIES EQ MAX
007043  013440  010000  017046          NOP                     %JNZRO  U.RV5Y  ;   THEN GO TO NEXT COPY (IF RCT)
007044  130541  000377  135620          ADDC    #377,R1         %CALL   S.STR1  ; RESET RVCRTY
007045  034446  000006  117054          CLR     R6              %JMP    U.RV5T  ; CLEAR R6/GO DO RETRY
                                ; ***   RETRIES FAILED - GO TO NEXT COPY ***
007046  010545  007003  135616  U.RV5Y: ADD     #SDI.SW,UBAR\N,BAR %CALL S.STRO ; [16K]SAVE MSCP STATUS
007047  013440  060000  107140  U.RV5W: NOP                     %JUPF   U.RVFH  ; [U52EC2]IF RBN THEN FATAL ERROR
007050  013740  007220  125604          MOV     #RVCRTY,BAR     %CALL   S.LLBO  ; [U52EC2]ERROR READING RCT/GET COPY COUNT
007051  031440  010000  157052          DEC\F   R0              %TZRO           ; IF EQ 0 THEN DON'T DECR
007052  013440  013000  147140          MOV\F   R0,BUF          %JZRO   U.RVFH  ; [16K]IF EQ 0 THEN FATAL ERROR
                                ; ***   RCT NEXT COPY RETRY CODE ***
007053  010545  007077  125600  U.RV5R: ADD     #SDI.RV,UBAR\N,BAR %CALL S.LDR6 ; GET SIZE OF RCT IN R6
007054                                  ASSUME  BUF.HH,EQ,BUF.HL+1              ; MAKE SURE SEQUENTIAL

                                ; ***   THIS POINT ENTERED TO CONTINUE FROM U.RV7A ***
007054  000547  007004  123211  U.RV5T: ADD     #BUF.HH,R7,Q,BAR %CALL  LDR2R1  ; R2,R1=LBN OF BLOCK IN ERROR OR FINAL SEEK
007055  013740  067250  156771          MOV\F   #RVECUP,BAR     %JUPF   U.RV3A  ; [UQA][U52EC2]BRANCH IF THE TARGET I/O IS BEING RETRIED
007056  017740  003051  000000          COM     #<-V.RVC3-1>&LOBYT,BUF          ; CHANGE uproc PROCESS VECTOR   [rae11]
```

```
007057  013740  007252  125602            MOV     #RVECDP,BAR     %CALL   S.LD10  ; CHANGE dproc PROCESS VECTOR   [rae11]
007060  031450  003010  106771            DEC     R10,R10,BUF     %JMP    U.RV3A  ; AND GO REREAD              [rae11]
                                ; ***   RCT SECTOR IS GOOD - SEARCH IT FOR REVECTOR LBN, EMPTY OR END ***
007061  034440  060000  117153  U.RV5Z:  CLR     R0              %JUPF   U.RV7A  ; [U52EC2]IF RBN THEN RETURN TO RV7A
                                                                                ; [U52EC2] OTHERWISE INIT RCT SEARCH DELTA
007062                                    ASSUME  RVCBNH,EQ,RVCBNL+1              ; MAKE SURE SEQUENTIAL
007062  003740  007131  133211            MOV     #RVCBNH,Q,BAR   %CALL   LDR2R1  ; R2/R1:HI/LO LBN TO REVECTOR
007063  010547  007002  135576            ADD     #BUF.BP,R7\N,BAR %CALL  S.LDR3  ; GET POINTER TO RCT DATA IN R3
007064  013740  007222  125600            MOV     #RCTOFF,BAR     %CALL   S.LDR6  ; GET INITIAL OFFSET TO SEARCH FROM IN R6
007065  133542  000040  010000            BIS     #RCT.PR,R2                      ; CONVERT LBN INTO RCT PRIMARY CODE, + CLR Z
007066  000146  010000  107101  U.RV5B:  ADD     R0,R6,Q         %JZRO   U.RV5D  ; BRANCH IF NO HITS, COMPUTE NEXT WORD OFFSET
007067  114640  000177  000000            BIT     #HIBYT-WRPFLG,Q                 ; [U52EC2] TEST FOR WORDPAIR OUT OF RANGE 0-376
007070  110043  017003  007076            ADDC    Q,R3\N,BAR      %JNZRO  U.RV5C  ; OFFSET INTO RCT, BR IF OFFSET OUT OF RANGE
007071  033710  000003  010000            MOV     (BUF),R10                       ; GET HIGH-ORDER WORD OF PAIR FIRST
007072                                    ASSUME  RCT.MT,EQ,0
007072                                    ASSUME  <-RCT.SE-1>&RCT.PR,EQ,0         ; TREAT PRIMARY & 2DARY CODES IN RCT THE SAME
007072  135550  010020  117114            BIC     #<RCT.SE-RCT.PR>,R10 %JZRO U.RVFE ; IF EMPTY THEN NO REPLACEMENT
007073                                    ASSUME  RCT.EN,EQ,100000
007073  016142  030010  117111            XOR     R10,R2\N        %JMSB   U.RV5F  ; BR IF END OF RCT, COMPARE HI WORDS
007074  010043  017003  047076            ADD\F   Q,R3\N,BAR      %JNZRO  U.RV5C  ; BR IF NOT EQUAL / GET LO RCT WORD
007075  016501  000003  010000            XOR     (BUF),R1\N                      ; COMPARE LOW WORDS
007076  132440  010000  107126  U.RV5C:  NEG     R0              %JZRO   U.RV5G  ; BRANCH IF MATCH, ADVANCE DELTA
007077  030560  030002  057100            ADD\T   #2,R0           %TNMSB          ; DELTA GOES 0,2,-2,4,-4,6,-6,...
007100  116540  000001  117066            XOR     #400,R0\N       %JMP    U.RV5B  ; UNTIL IT HITS 256 (BLOCK IS EXHAUSTED)

                                ; ***   THIS RCT BLOCK IS EXHAUSTED - GO TO THE NEXT ONE ***
007101  013740  007224  125574  U.RV5D:  MOV     #RCTBLK,BAR     %CALL   S.LDR1  ; GET THE RCT BLOCK OFFSET
007102  130441  000001  135620            INC     R1              %CALL   S.STR1  ; AND INCREMENT IT
007103  013740  007222  000000  U.RV5E:  MOV     #RCTOFF,BAR                     ; POINT TO INTRA-BLOCK OFFSET WORD IN MEMORY
007104  134546  003200  010000            AND     #WRPFLG,R6,BUF                  ; [U52EC2]ZERO OFFSET WHILE PRESERVING WRAP FLAG
007105  013740  007250  000000            MOV     #RVECUP,BAR                     ; [UQA]BAR:PTR TO PROCESS VECTOR WORD/RET
007106  017740  003051  000000            COM     #<-V.RVC3-1>&LOBYT,BUF          ; CHANGE uproc PROCESS VECTOR   [rae11]
007107  013740  007252  125602            MOV     #RVECDP,BAR     %CALL   S.LD10  ; CHANGE dproc PROCESS VECTOR   [rae11]
007110  031450  003010  116762            DEC     R10,R10,BUF     %JMP    U.RVC3  ; AND GO REREAD                 [rae11]
                                                                                ; [U52EC2] BACK TO RCT READ STEP[rae11]

                                ; ***   END OF RCT DETECTED - WRAP BACK TO BEGINNING, BUT ONLY ONCE... ***
007111  013740  007224  125630  U.RV5F:  MOV     #RCTBLK,BAR     %CALL   S.CLRB  ; CLEAR BLOCK OFFSET WITHIN RCT
007112  114546  000200  000000            BIT     #WRPFLG,R6                      ; SEE IF WE HAVE DONE THIS ALREADY
007113  133546  010200  107103            BIS     #WRPFLG,R6      %JZRO   U.RV5E  ; IF NOT, CONTINUE AT BLOCK 0 OF RCT
007114  013740  007132  125573  U.RVFE:  MOV     #RVCTA,BAR      %CALL   S.LDR0  ; [EERREC]GET REAL TIME COMMAND
007115  134540  000037  010000            AND     #17400,R0                       ; [EERREC]MASK IT
007116  136540  000027  010000            XOR     #READCD,R0                      ; [EERREC]WAS IT A READ?
007117  033760  010110  047141            MOV\T   #<ST.DAT!SC.HDR>,R0 %JNZRO U.RVFB ; [EERREC]IF NOT, BRANCH
007120  010547  007001  135573            ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR0  ; [EERREC]ELSE, GET STATUS
007121  033540  003040  010000            BIS     #BRCTS,R0,BUF                   ; [EERREC]SET RCT SEARCHED AND EXIT
007122  013740  007250  000000            MOV     #RVECUP,BAR                     ; [EERREC]NOW SKIP STEP 6 FOR U
007123  017740  003047  010000            COM     #<<-V.RVC7-1>&LOBYT>,BUF        ; [EERREC]
007124  013740  007252  135574            MOV     #RVECDP,BAR     %CALL   S.LDR1  ; [EERREC]SKIP STEP 6 FOR D
007125  130441  003001  127777            INC     R1,R1,BUF       %RTN            ; [EERREC]

                                ; ***   COME HERE IF WE ACTUALLY FIND THE DESIRED LBN IN THE RCT (GLORY BE!!) ***
                                ; ***   Q = WORD OFFSET INTO RCTBLK ***
007126  054643  000376  000000  U.RV5G:  AND\R   #376,Q,R3                       ; CONVERT WORD OFFSET INTO ITEM OFFSET
```



```
007127  013740  007224  125574            MOV     #RCTBLK,BAR     %CALL   S.LDR1  ; GET RCT BLOCK OFFSET
007130  003740  000011  125501            MOV     #9.,Q           %CALL   S.RRR1  ; MULTIPLY BY 128
007131  005541  000177  000000            BIC     #177,R1,Q                       ; ISOLATE LOW-ORDER PRODUCT IN Q
007132  036042  000001  000000            XOR     R1,Q,R2                         ; AND HIGH-ORDER PRODUCT IN R2
007133  133542  000140  137151            BIS     #RBNCOD,R2      %CALL   U.RVRT  ; [ECO#1]ADD RBN HEADER TYPE TO HI RBN/RESET BUF.TA
007134  033041  000003  010000            BIS     R3,Q,R1                         ; [U52EC2][ECO#1]COMBINE ITEM OFFSET INTO LO RBN
007135  010545  007003  125630            ADD     #SDI.SW,UBAR\N,BAR %CALL S.CLRB ; [U52EC3]ZAP ERROR CODE (IF ANY)
007136  010545  007067  125573            ADD     #SDI.RT,UBAR\N,BAR %CALL S.LDR0 ; GET RBN'S/TRACK IN R0<5:0>
007137  034540  000077  116774            AND     #RBNMSK,R0      %JMP    U.RV3C  ; ISOLATE RBN'S/TRACK IN R0
                                .PAGE
```

```
                                    ; ***   ERROR ROUTINES AND SUBROUTINES ***
                                    ; ***   FATAL ERROR EXIT (RBN OR RCT LBN) ***
007140  010545  007003  135573  U.RVFH: ADD     #SDI.SW,UBAR\N,BAR %CALL S.LDRO ; RO=FATAL ERROR STATUS
007141  013440  000000  127147  U.RVFB: NOP                       %CALL   U.RVRL  ; [U52EC2][ECO#1]GET ORIG LBN IN BUF.HL/BUF.HH
007142  010545  007026  135574          ADD     #SDI.ES,UBAR\N,BAR %CALL S.LDR1 ; [U52EC2]R1=EXTENDED STATUS
007143  033541  003020  134574          BIS     #RVACTV,R1,BUF  %CALL   U.DERE  ; [U52EC2][16K]SET REVECT ACTIVE/GO DO LOGS,ETC
007144  013740  007254  135574  U.RVFA: MOV     #RVCFLG,BAR     %CALL   S.LDR1  ; [UQA]R1=PROC FLAG
007145  037441  000001  000000          COM     R1                              ; [UQA]SWITCH TO D.PROC
007146  135541  003200  137777          BIC     #BIT15,R1,BUF   %RET            ; [UQA]CLEAR ERROR BIT/RETURN

                                    ; ***   RESTORE ORIGINAL LBN TO BUFFER CONTROL AREA POINTED TO BY R7 ***
007147                              U.RVRL: ASSUME  <RVCBNH&17>,EQ,<RVCBNL&17>+1    ; MAKE SURE AUTO INCR WILL WORK
007147  003740  007131  133211          MOV     #RVCBNH,Q,BAR   %CALL   LDR2R1  ; R2,R1=SAVED ORIGINAL LBN
007150                              ASSUME  BUF.HH,EQ,BUF.HL+1              ; MAKE SURE SEQUENTIAL
007150  000547  007003  113213          ADD     #BUF.HL,R7,Q,BAR %JMP   STR1R2  ; RESTORE ORIGINAL LBN/RETURN

                                    ; ***   RESTORE ORIGINAL BUF.TA ***
007151  013740  007132  125573  U.RVRT: MOV     #RVCTA,BAR      %CALL   S.LDRO  ; RO=ORIGINAL BUF.TA
007152  010547  007005  115616          ADD     #BUF.TA,R7\N,BAR %JMP   S.STRO  ; RESTORE TO BUFFER/RETURN
                                        .PAGE
```



```
                                    ;       WE COME HERE WHEN THE OPERATION ON THE REVECTOR BLOCK IS COMPLETE, TO CHECK FOR ERRORS
                                    ;       AND EITHER RETRY OR POSITION BACK INTO THE STREAM OF USER I/O

                                    U.RVC7: ;NOP            @SUPF   %JMP    U.RV57  ; [U52EC2]ACTUALLY DONE IN DISPATCH VECTOR...

                                    ; ***   U.RV57 RETURNS CONTROL AT U.RV7A IF EVERYTHING IS KOSHER ***
                                    ; ***   SUCCESS AT LAST!!! - WE HAVE PERFORMED THE ORIGINAL I/O ON THE RBN ***

007153  034446  000006  137151  U.RV7A: CLR     R6                      %CALL   U.RVRT  ; RESTORE BUF.TA (IN CASE RETRY)
007154  013740  007230  135573          MOV     #RVCEBF,BAR             %CALL   S.LDRO  ; RESTORE BUFFER STATUS
007155  136540  000100  127147          XOR     #BFULL,RO               %CALL   U.RVRL  ; FLIP FULL FLAG/RESTORE ORIGINAL LBN
007156  010547  007000  125616          ADD     #BUF.NL,R7\N,BAR %CALL  S.STRO  ; THIS SETS BECC=0 AND RESTORES OLD BLAST
007157  010547  007001  125574          ADD     #BUF.ST,R7\N,BAR %CALL  S.LDR1  ; [cho8] LOOK AT THE STATUS
007160  014541  000040  000000          BIT     #BRCTS,R1                       ; [cho8] SEE IF LBN NOT IN RCT
007161  013440  010000  017054          TST     RO                      %JNZRO U.RV5T   ; [cho8] IF SET, BYPASS SDI.DB UPDATE
007162                              ASSUME  BLAST,EQ,BIT15                  ; MAKE SURE BLAST IS MSB FOR CHECK
007162  010545  037007  035616          ADD     #SDI.DB,UBAR\N,BAR %CNMSB S.STRO; [U52EC2]IF NOT LAST THEN ADVANCE
                                                                            ; [U52EC2]BUFR PTR PAST REVECTORED BUF
                                    ; ***   SEEK BACK TO THE USER'S OLD CYLINDER - @SUPF IS SET TO INDICATE THIS.
007163  013440  000000  117054          NOP                             %JMP    U.RV5T  ; GO FAKE AN ERROR RETRY TO POSITION
                                                                                        ; THE DISK THE SAME AS WHEN THE REVECTOR STARTED

                                    .page
```

```
                                ;       NOTE:   This 3 word (word = 48 bits) pattern is used by test
                                ;               engineering.  It should never be moved.  The point
                                ;               is to have an all 1 and an all zero bit pattern for
                                ;               each of the six bytes.  This is complicated by the
                                ;               fact that parity must be maintained across the 48 bits
                                ;               as diagnostics will fail even if these are not executed.
                                ;               Byte 0 is the least significant byte in my comments below.
                                .org    7375
007375  177777  177777  177401  .word   177777,177777,177401    ; 1s in bytes 5,4,3,2,1 (not 0)
                                .org    7376
007376  000000  000001  000000  .word   000000,000001,000000    ; 0s in bytes 5,4,3,  1,0 (not 2)
                                .org    7377
007377  000000  000000  000777  .word   000000,000000,000777    ; 1 in byte 0, 0 in byte 2
```



```
                                .SBTTL  HARDWARE/SOFTWARE VECTOR TABLES

                                .ORG    ^HOF00
007400                          .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO ERROR ROUTINE

                                .ORG    ^HOF03
                                ;;*****************************************************************************
                                ;
                                ;       THE PAR (Pipeline Address Register) IS FORCED HERE TO THE LO VECTOR
                                ;       TABLE AND TO THE HI VECTOR TABLE DURING THE PAR TEST.
                                ;
                                ;       NOTE:   THE LOCATION LOPAR (^HOF03) AND HIPAR (^HOFFC), WHICH ARE USED
                                ;               IN THE PAR TEST, MUST NOT BE CHANGED.
                                ;
                                ;;*****************************************************************************

007403  013440  000000  117774  LOPAR:  NOP                     %JMP    HIPAR   ;JUMP TO HIPAR

                                ;;*****************************************************************************
                                ;
                                ;       Read transfer vector table
                                ;
                                ;;*****************************************************************************
                                ;       Transfer table for ultra-optimized read                                 ;[E121]
                                BIRDBR:                                                                         ;[E121]
007404  131545  000010  115703          SUB     #8.,UBAR        %JMP    BIRD00  ;Predecr UBAR, enter proper loop ;[E121]
007405  131545  000010  105720          SUB     #8.,UBAR        %JMP    BIRD01  ;Predecr UBAR, enter proper loop ;[E121]
007406  131545  000010  115735          SUB     #8.,UBAR        %JMP    BIRD02  ;Predecr UBAR, enter proper loop ;[E121]
007407  131545  000010  105752          SUB     #8.,UBAR        %JMP    BIRD03  ;Predecr UBAR, enter proper loop ;[E121]
007410  131545  000010  105767          SUB     #8.,UBAR        %JMP    BIRD04  ;Predecr UBAR, enter proper loop ;[E121]
007411  131545  000010  116004          SUB     #8.,UBAR        %JMP    BIRD05  ;Predecr UBAR, enter proper loop ;[E121]
007412  131545  000010  106021          SUB     #8.,UBAR        %JMP    BIRD06  ;Predecr UBAR, enter proper loop ;[E121]
007413  131545  000010  106036          SUB     #8.,UBAR        %JMP    BIRD07  ;Predecr UBAR, enter proper loop ;[E121]
007414  131545  000010  115676          SUB     #8.,UBAR        %JMP    BIRD08  ;Predecr UBAR, enter proper loop ;[E121]
007415  131545  000010  115712          SUB     #8.,UBAR        %JMP    BIRD09  ;Predecr UBAR, enter proper loop ;[E121]
007416  131545  000010  115727          SUB     #8.,UBAR        %JMP    BIRD10  ;Predecr UBAR, enter proper loop ;[E121]
007417  131545  000010  115744          SUB     #8.,UBAR        %JMP    BIRD11  ;Predecr UBAR, enter proper loop ;[E121]
007420  131545  000010  105761          SUB     #8.,UBAR        %JMP    BIRD12  ;Predecr UBAR, enter proper loop ;[E121]
007421  131545  000010  105776          SUB     #8.,UBAR        %JMP    BIRD13  ;Predecr UBAR, enter proper loop ;[E121]
007422  131545  000010  116013          SUB     #8.,UBAR        %JMP    BIRD14  ;Predecr UBAR, enter proper loop ;[E121]
007423  131545  000010  106030          SUB     #8.,UBAR        %JMP    BIRD15  ;Predecr UBAR, enter proper loop ;[E121]
                                                                                                                ;[E121]
                                ;       ************** Pop mechanism    ***************
                                ;       This is the address #POP plus PAR 74xx (octal).  This is
                                ;       used to unwrap the stack from a timeout error condition.

007424  013440  000000  127777  POP1:   NOP                     %RET            ; this return is taken

                                ;       Pop mechanism.  This is used to unwrap the stack when an invalid
                                ;       PTE is detected.
```

```
007425  013440  000000  115312  POP2:   NOP                     %JMP    U.INVA  ; drop u.getp ret

                                .page
```



```
                                ;;******************************************************************************
                                ;
                                ; TABLE NAME:
                                ;       UTSTBL
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       This table contains the addresses of U-PROC tests that can be looped
                                ;       on.
                                ;
                                ;;******************************************************************************

000026                          TESTBL  = . - 7400

007426  034447  000007  110077  UTSTBL: CLR     UER             %JMP    T0      ;LOOP U-PROC TEST
007427  034447  000007  110102          CLR     UER             %JMP    T1      ;LOOP U-PROC TEST
007430  034447  000007  110132          CLR     UER             %JMP    T2      ;LOOP U-PROC TEST
007431  034447  000007  100136          CLR     UER             %JMP    HU.T3   ;HANG U-PROC, LET D-PROC LOOP
007432  034447  000007  100136          CLR     UER             %JMP    HU.T4   ;HANG U-PROC, LET D-PROC LOOP
007433  034447  000007  100136          CLR     UER             %JMP    HU.T5   ;HANG U-PROC, LET D-PROC LOOP
007434  034447  000007  100136          CLR     UER             %JMP    HU.T6   ;HANG U-PROC, LET D-PROC LOOP
007435  034447  000007  110145          CLR     UER             %JMP    HU.T7   ;HANG U-PROC, LET D-PROC LOOP
007436  034447  000007  110146          CLR     UER             %JMP    T8      ;LOOP U-PROC TEST
007437  034447  000007  110152          CLR     UER             %JMP    HU.T9   ;HANG U-PROC, LET D-PROC LOOP
007440  033701  000003  010000  SETHVR: MOV     (BUF),R1                        ; [mjt06] get the hardware revision (in lower byte)
007441  033243  000003  010000          MOV     Q,R3                            ; [mjt06] save q
007442  003740  000010  135501          MOV     #8.,Q           %CALL   S.RRR1  ; [mjt06] rotate r1 to upper byte
007443  003443  000003  010000          MOV     R3,Q                            ; [mjt06] restore q
007444  033140  000001  103370          OR      R1,R0           %JMP    STROLG  ; [mjt06] Get the hardware revision number and stuff it into
                                .page
```

```
                                ;++
                                ; TABLE NAME:
                                ;       VECTAB  (CONTROLLER HARDWARE ERROR, STATE AND ERROR VECTOR TABLE)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS TABLE CONTAINS ALL OF THE VECTOR TABLES FOR BOTH THE U.PROC
                                ; AND D.PROC. CONTROLLER MICROCODE LOADING THE 'PAR' REGISTER IN PREPARATION FOR
                                ; ENTERING THIS TABLE NEED ONLY LOAD THE LEAST SIGNIFICANT 8 BITS (LOW BYTE)
                                ; OF THE REQUIRED OFFSET, THE CONTROLLER HARDWARE WILL FORCE THE MOST SIGNIFICANT
                                ; 8 BITS (HI BYTE) TO ALL ONES. THE CONTROLLER ERROR DETECTING HARDWARE USES THE
                                ; LAST 16 ENTRIES OF THE VECTOR TABLE AND THESE ENTRIES MUST START AT ADDRESS
                                ; OFF0 HEX). A DESCRIPTION OF HOW THE HARDWARE FORMS THE VECTOR ADDRESS
                                ; AND THE ERRORS DETECTED FOLLOW:
                                ;
                                ;       ERROR VECTOR ADDRESS - GENERAL
                                ;               THE ERROR VECTOR ADDRESS IS 'BIT ENCODED' BASED ON THE ERROR
                                ;               WHICH OCCURRED.   IN ALL CASES, THE INSTRUCTION IN WHICH THE
                                ;               ERROR OCCURRED WILL COMPLETE, THE FOLLOWING INSTRUCTION WILL
                                ;               COMPLETE, AND THE NEXT INSTRUCTION WILL BE THE ERROR VECTOR.
                                ;
                                ;               SEQ ADR BIT 03 OCTAL DECODE;
                                ;                       0= D PROCESSOR
                                ;                       1= U PROCESSOR
                                ;               SEQ ADR BITS 02-00 OCTAL DECODE;
                                ;                       0= NOT USED BY HARDWARE
                                ;                       1= BI READ PARITY ERROR 2
                                ;                       2= BI WRITE PARITY ERROR 2
                                ;                       3= BCI PARITY ERROR 1
                                ;                       4= NOT USED BY HARDWARE
                                ;                       5= RAM PARITY ERROR
                                ;                       6= TIME OUT ERROR
                                ;                       7= CROM PARITY ERROR
                                ;
                                ;       CONTROL MEMORY PARITY ERROR
                                ;               THE  PROCESSOR  WITH THE  CROM PARITY ERROR  WILL  EXECUTE  THE
                                ;               FAULTY INSTRUCTION, THE FOLLOWING INSTRUCTION, THEN  THE  ERROR
                                ;               VECTOR INSTRUCTION.  THE SIGNAL 'CROM PE L'  WILL BECOME ACTIVE
                                ;               ON THE CYCLE FOLLOWING THE ERROR. THIS CORRESPONDS TO THE THIRD
                                ;               INSTRUCTION AFTER THE ERROR WHEN USING THE LOGIC ANALYSER.
                                ;               IF THE VECTOR ADDRESS HAS A PARITY ERROR, THE PROCESSOR
                                ;               WILL HANG. (UNTIL IT GOES AWAY)  SO DISPLAY STATUS.
                                ;
                                ;       TIMEOUT ERROR
                                ;               THE TIMEOUT CIRCUIT IS ENABLED WHEN EVER A SUCCESSFUL JUMP IS
                                ;               EXECUTED. FALLING THROUGH TO THE NEXT INSTRUCTION RESETS THE
                                ;               TIMEOUT CIRCUIT. NOTE - TIMEOUT ERRORS DO NOT OCCUR ON THE
                                ;               UNIBUS PROCESSOR.
                                ;
                                ; ****THE HARDWARE MUST BE CHANGED TO DISABLE THE UNIBUS TIMEOUT.*****
                                ;
                                ;TIME=  <16600.*7/1000.>                ; MSEC MINIMUM TIMEOUT
                                ;TIME=  <16600.*8/1000.>                ; MSEC MAXIMUM TIMEOUT
                                ;INST=  <16600.*7*1000./INSTR>          ; INSTR MINIMUN
```



```
                                ;--

                                ; ***   NULL AREA WITH JUMP TO ERROR ROUTINE (+PARITY) - JUST IN CASE ***

007445                                  .FILL   13440,0,<BIT15+BIT12+SEQERR>    ;JUMP TO ERROR ROUTINE

                                .ORG    ^HOFD5                          ; ORIGIN OFD5 HEX
                                D.STPA:                                 ; FOR THE LAST .IF

                                ; ***   NOTE - IF VECTAB SIZE CHANGES THEN THE PRECEEDING .ORG MUST ALSO ***
                                ; ***   THE EQUATION IS ORG.VALUE = FE9H - SIZE OF VECTAB ***
                                ; ***   START CONTROLLER VECTOR TABLE ***

                                ; ***   REVECTORING STATE VECTORS - MUST START ON ODD WORD.
                                ;       CODE IN VECTOR IS TO STORE INCREMENTED VECTOR PTR BACK IN MEMORY
007725                          ;       ASSUME  .&1,EQ,1                        ; UPROC STARTS FIRST.....
                                VECTAB:
                                V.URVC:
007725  013440  003001  107020  V.RVCT: MOV     R1,BUF          %JMP    U.RVUX  ; [UQA]SYNCHRONIZE UPPER PROCESSOR
007726  013440  000000  116762  V.RVC3: NOP                     %JMP    U.RVC3  ; INITIATE READ OF RCT BLOCK
007727  013440  000600  107022  V.RVC5: NOP             @RUPF   %JMP    U.RVC5  ; [US2EC2]SEARCH RCT BLOCK FOR REPLACEMENT
007730  013440  000400  117022  V.RVC7: NOP             @SUPF   %JMP    U.RV57  ; [US2EC2]CHECK OPERATION FOR ERRORS & RETRY
                                V.RVC8:                                         ; FOR LAST STEP

                                ; ***   DUP VECTOR TABLE ***

007731  000550  000007  113725  STDTBL: ADD     #<P.BCNO-1>,R10,Q %JMP  U.GSST  ; 1=GET DUST STATUS COMMAND
007732  114544  000040  103737          BIT     #DMBEG,RLL      %JMP    U.EXSP  ; 2=EXECUTE SUPPLIED PROGRAM
007733  033740  000001  102420  U.STER: MOV     #ST.CMD,R0      %JMP    U.C2EP  ; 3=EXECUTE LOCAL PROGRAM COMMAND
007734  013740  007150  103763          MOV     #MWQUE,BAR      %JMP    U.DATA  ; 4=SEND DATA COMMAND
007735  013740  007146  113763          MOV     #MRQUE,BAR      %JMP    U.DATA  ; 5=RECEIVE DATA COMMAND
007736  135544  000060  102303          BIC     #<DXFC!DMBEG>,RLL %JMP  U.IMEX  ; [16K]6=ABORT PROGRAM COMMAND
000006                          MAXSTC  = .-STDTBL

                                ; ***   UNIBUS PROCESSOR XFC VECTORS ***

                                XFCTAB:
000012                          XFCUPR := .-XFCTAB+12

007737  033751  000001  115407  UP.XFC: MOV     #1,R11          %JMP    UXFC11  ; XFC 11 - READ UNIBUS MEMORY
007740  034451  000011  115407          CLR     R11             %JMP    UXFC11  ; XFC 12 - WRITE UNIBUS MEMORY
007741  013740  007007  115423          MOV     #DMREGO,BAR     %JMP    UXFC13  ; XFC 13 - PERFORM ECC CORRECTION
007742  033747  007146  115427          MOV     #MRQUE,R7,BAR   %JMP    UXF14A  ; XFC 14 - SEND DATA
007743  033747  007150  105427          MOV     #MWQUE,R7,BAR   %JMP    UXF14A  ; XFC 15 - RECEIVE DATA
007744  033743  000005  105446          MOV     #5,R3           %JMP    UXFC16  ; XFC 16 - CONVERT TO PHYSICAL ADDRESS
007745  033740  000026  115454          MOV     #<16384.-DM.BEG>&LOBYT,R0 %JMP UXFC17; [16K]XFC 17 - EXIT DM MODE
000021                          XFCFIN  := .-UP.XFC+XFCUPR                      ; XFC DONE VALUE

007746  033740  000360  105462          MOV     #DM.BEG+DSTSL&LOBYT,R0 %JMP UXFC18 ; [16K]XFC 18 - INCREMENT DUST STATUS
000022                          XFCMAX  := .-UP.XFC+XFCUPR                      ; MAXIMUM XFC VALUE

007747                                  ASSUME XFCMAX,EQ,22
007747                                  ASSUME XFCFIN,EQ,21
007747                                  ASSUME XFCUPR,EQ,12
```

```
007747  013440  000000  000000          NOP                                     ; FOR ALIGNMENT
                                .ORG    ^HOFE8                                  ; *** IMPORTANT ***

                                ; ***   THE U.HWVR SUBROUTINE MUST START AT OFE8 ***
                                ; ***   THE U.SCID SUBROUTINE MUST START AT OFE9 ***

                                ;;+
                                ; ROUTINE NAME:
                                ;       U.HWVR  (SET HARDWARE VERSION NUMBER)
                                ;       U.SCID  (SET CONTROLLER ID)
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THE U.HWVR ROUTINE WILL SET THE CONTROLLER HARDWARE VERSION NUMBER IN
                                ; AREA POINTED TO BY Q. THIS ROUTINE STARTS AT 7750 OCTAL AND ALL HARDWARE
                                ; VERSION NUMBERS IN THE 8 BIT LITERAL FIELD MUST HAVE EVEN PARITY.
                                ;       THE U.SCID ROUTINE WILL SET THE CONTROLLER IDENTIFIER IN THE AREA
                                ; POINTED TO BY REGISTER Q. THIS ROUTINE STARTS AT 7751 OCTAL AND
                                ; ALL VALUES PLACED IN THE 8 BIT LITERAL FIELDS MUST HAVE EVEN PARITY.
                                ;
                                ; INPUTS:
                                ;       Q               POINTER TO STORAGE AREA
                                ;
                                ; OUTPUTS:
                                ;       U.SCID  CONTROLLER ID IN AREA POINTED TO BY Q
                                ;       U.HWVR  CONTROLLER HARDWARE VERSION NUMBER IN AREA POINTED TO BY Q
                                ;
                                ;;-

007750  013740  007275  117440  U.HWVR: MOV     #BUFP49,BAR     %JMP    SETHVR  ; GO STORE SW/HW VERSION #/RETURN
007751  033740  000000  010000  U.SCID: MOV     #0,R0                           ; R0=LO BYTE OF 1ST WORD
007752  133540  000000  123370          OR      ##0,R0          %CALL   STROLG  ; OR IN HI BYTE/STORE
007753  033740  000000  010000          MOV     #0,R0                           ; R0=LO BYTE OF 2ND WORD
007754  133540  000000  123370          OR      ##0,R0          %CALL   STROLG  ; OR IN HI BYTE/STORE
007755  033740  000000  010000          MOV     #0,R0                           ; R0=LO BYTE OF 3RD WORD
007756  133540  000000  123370          OR      ##0,R0          %CALL   STROLG  ; OR IN HI BYTE/STORE
007757  133740  000001  115477          MOV     #CCLASS,R0      %JMP    U.MODL  ; R0=CONTROLLER CLASS jump to insert model [rae04]

                                ; ***   NOTE - THESE HARDWARE ERROR VECTORS MUST START AT OFFOH ***
                                ; ***   D.PROCESSOR ERROR VECTORS ***

                                .ORG    ^HOFF0                                  ; *** IMPORTANT ***
007760  033447  000004  117761  E.WPE2: MOV     CRI,R7          %JMP    .+1     ; 0 - NOT USED (BY HARDWARE)
007761  134547  000004  107762          AND     #DFAIL,R7       %JMP    .+1     ; 1 - NOT USED (BY HARDWARE)
007762  133547  004020  117763          BIS     #ER.SRP,R7,UCRD %JMP    .+1     ; 2 - NOT USED (BY HARDWARE)
007763  174547  000020  117767          AND\L   #ER.SRP,R7      %JMP    U.ER    ; 3 - NOT USED (BY HARDWARE)
007764  173747  004020  107767  E.RPE2: MOV\L   #ER.SRP,R7,UCRD %JMP    U.ER    ; 4 - NOT USED (BY HARDWARE)
007765  114544  000002  137777          BIT     #INDIAG,RLL     %RET            ; 5 - NOT USED (BY HARDWARE)
007766  170447  000007  107765  CON.ER: INC\L   R7,R7            %JMP    .-1     ; 6 - NOT USED (BY HARDWARE)
007767  070147  000007  107770  U.ER:   ADD\L   R7,R7            %JMP    .+1     ; 7 - NOT USED (BY HARDWARE)

                                ; ***   U.PROCESSOR ERROR VECTORS ***

007770  170447  000007  102067          INC\L   R7,R7            %JMP    U.HERR  ; 8 - NOT USED (BY HARDWARE)
```



```
007771  015546  006001  107764          BIC     #B.NABO,R6\N,BREG %JMP  E.RPE2  ; 9 - BI RD PE2          [mr1005]
007772  015546  006001  117760          BIC     #B.NABO,R6\N,BREG %JMP  E.WPE2  ; A - BI WR PE2          [mr1005]
007773  173747  004020  107767          MOV\L   #ER.SRP,R7,UCRD %JMP    U.ER    ; B - BCI PE1
007774  070147  000627  107766  HIPAR:  ADD\L   R7,R7 @SCLR     %JMP    CON.ER  ; C - NOT USED (BY HARDWARE)
007775  033707  003003  117776          MOV     (BUF),R7,BUF    %JMP    .+1     ; D - RAM PE (ATTEMPT TO CLEAR ERROR)
007776  173747  004040  107767          MOV\L   #ER.SAP,R7,UCRD %JMP    U.ER    ; E - NOT USED (BY HARDWARE)
007777  173747  004060  117767          MOV\L   #ER.SOP,R7,UCRD %JMP    U.ER    ; F - CROM PE

010000                                  .END            ;END OF FILE
```

A	006604	ACLO	000007	ALGADR	034000	ALGERR	000071	ALOLMT	000035
ALUTST	000373	ATTCOD	000100	ATTN	000002	AVAIL	000100	AVL.LN	000040
B	006611	BA	037777	BADRH	000266	BADRL	000265	BAD.LN	000034
BANDT	140000	BAR	000007	BA16	000001	BA17	000002	BA18	000004
BA19	000010	BA20	000020	BA21	000040	BA22	000100	BA23	000200
BA24	000400	BA25	001000	BA26	002000	BA27	004000	BA28	010000
BA29	020000	BCAID	000005	BCAIS	000015	BCATST	000530	BCGRP	000002
BCICSR	000050	BDCST	000015	BDSNF	002000	BECC	010000	BECER	020000
BERDN	000400	BERMAX	000150	BFRQ	000020	BFSV	000100	BFULL	040000
BGOOD	004000	BGRUP	000001	BIBAD	000037	BIBER	000010	BICSR	000004
BICSRE	000200	BIDTYP	000000	BIEADR	000044	BIECSR	000014	BIGPRO	000360
BIGPR1	000364	BIGPR2	000370	BIGPR3	000374	BIIDES	000020	BIIMSK	000024
BIITST	000751	BIMEM	001026	BIPE	000630	BIPSDE	000030	BIPSFC	000060
BIPSRC	000034	BIRDBR	007404	BIRDCM	000001	BIRD00	005703	BIRD01	005720
BIRD02	005735	BIRD03	005752	BIRD04	005767	BIRD05	006004	BIRD06	006021
BIRD07	006036	BIRD08	005676	BIRD09	005712	BIRD10	005727	BIRD11	005744
BIRD12	005761	BIRD13	005776	BIRD14	006013	BIRD15	006030	BIREOK	006112
BIRTRY	000300	BIR.CM	006060	BIR.EV	006053	BIR.EX	006075	BIR.LP	006041
BIR.LW	006061	BIR.L2	006052	BIR.NI	006047	BIR.NO	006113	BIR.OD	006042
BIR.QK	006062	BIR.SO	006114	BIR.X2	006076	BIR.1W	006074	BIR.2W	006073
BIR.3W	006072	BIR.4W	006071	BIR.5W	006070	BIR.6W	006067	BISADR	000040
BIT00	000001	BIT01	000002	BIT02	000004	BIT03	000010	BIT04	000020
BIT05	000040	BIT06	000100	BIT07	000200	BIT08	000400	BIT09	001000
BIT10	002000	BIT11	004000	BIT12	010000	BIT13	020000	BIT14	040000
BIT15	100000	BIT16	000001	BIT17	000002	BIT18	000004	BIT19	000010
BIT20	000020	BIT21	000040	BIT22	000100	BIT23	000200	BIT24	000400
BIT25	001000	BIT26	002000	BIT27	004000	BIT28	010000	BIT29	020000
BIT30	040000	BIT31	100000	BIUCSR	000100	BIWRCM	000004	BIWSTA	000054
BIW.EX	006325	BIW.LP	006274	BIW.LQ	006250	BIW.LS	006330	BIW.NO	006417
BIW.QK	006302	BIW.SO	006361	BIW.TS	006326	BIW.5R	006312	BI.ERR	006210
BI.EX	006231	BI.RD	005635	BI.RLP	005643	BI.STO	006235	BI.STP	006237
BI.TO	006201	BI.UDO	006303	BI.UDX	006065	BI.WOW	006407	BI.WR	006242
BLAST	100000	BLOCK	000001	BLRWR	001000	BLSTB	040000	BMAPDN	000010
BNXCOP	000004	BOARD2	001000	BP	006614	BRARS	000020	BRCTS	000040
BREG	000006	BRGADR	000265	BRLV4	000001	BRLV4R	000400	BRLV4S	000020
BRLV5	000002	BRLV5R	001000	BRLV5S	000040	BRLV6	000004	BRLV6R	002000
BRLV6S	000100	BRLV7	000010	BRLV7R	004000	BRLV7S	000200	BROKE	010000
BRTRY	100000	BROOLP	005672	BROOLX	005673	BRO1LP	005706	BRO1LX	005707
BRO2LP	005723	BRO2LX	005724	BRO3LP	005740	BRO3LX	005741	BRO4LP	005755
BRO4LX	005756	BRO5LP	005772	BRO5LX	005773	BRO6LP	006007	BRO6LX	006010
BRO7LP	006024	BRO7LX	006025	BRO8LP	005677	BRO9LP	005713	BR10LP	005730
BR11LP	005745	BR12LP	005762	BR13LP	005777	BR14LP	006014	BR15LP	006031
BTCNT	000236	BTROTH	004000	BUF	000003	BUFBEG	002055	BUFEND	005255
BUFLMT	000041	BUFPTR	000133	BUFP00	000133	BUFP01	000135	BUFP02	000137
BUFP03	000141	BUFP04	000143	BUFP05	000145	BUFP06	000147	BUFP07	000151
BUFP08	000153	BUFP09	000155	BUFP10	000157	BUFP11	000161	BUFP12	000163
BUFP13	000165	BUFP14	000167	BUFP15	000171	BUFP16	000173	BUFP17	000175
BUFP18	000177	BUFP19	000201	BUFP20	000203	BUFP21	000205	BUFP22	000207
BUFP23	000211	BUFP24	000213	BUFP25	000215	BUFP26	000217	BUFP27	000221
BUFP28	000223	BUFP29	000225	BUFP30	000227	BUFP31	000231	BUFP32	000233
BUFP33	000235	BUFP34	000237	BUFP35	000241	BUFP36	000243	BUFP37	000245
BUFP38	000247	BUFP39	000251	BUFP40	000253	BUFP41	000255	BUFP42	000257
BUFP43	000261	BUFP44	000263	BUFP45	000265	BUFP46	000267	BUFP47	000271
BUFP48	000273	BUFP49	000275	BUFP50	000277	BUFP51	000301	BUFP52	000303
BUFR1.	005255	BUFR2.	005672	BUF.BC	000012	BUF.BP	000002	BUF.DL	000415
BUF.EC	000401	BUF.ED	000400	BUF.GP	000011	BUF.HH	000004	BUF.HL	000003
BUF.LL	000015	BUF.NL	000000	BUF.SD	000006	BUF.ST	000001	BUF.TA	000005
BUF.UA	000007	BUF.US	000010	BUF.U1	000013	BUF.U2	000014	BUF.56	000000
BW.SO1	006367	B.BAD	000010	B.BTO	000004	B.CPE	000200	B.CTE	010000
B.DTYP	000416	B.ICE	000001	B.ISE	002000	B.IVE	000400	B.LED	000020
B.LOOP	000100	B.MPE	004000	B.MTCE	020000	B.NABO	000001	B.NEX	000002

B.NMR	040000	B.NRTY	000200	B.ODD	000040	B.RDS	000040	B.RTO	000020
B.SPE	000100	B.STO	000010	B.TDF	001000	CCLASS	000400	CCSRWO	000032
CCSRW1	000033	CDONE	000016	CFLAGS	000320	CF.ATN	000200	CF.MSC	000100
CF.THS	000020	CHGFLG	000202	CHGMOD	000201	CHKBDR	006170	CHKPAR	000353
CKWREA	004124	CKWRER	004123	CKWROM	004121	CLIMIT	000142	CLRERR	000251
CLRTST	000075	CMDCOF	000030	CMDLEN	000026	CMDLIM	000024	CMDO	000004
CMDPOF	000031	CMDPTR	000027	CMODEL	000022	CMPBUF	000242	CNTFLG	000170
CNT.LN	000030	CNVTV1	000040	CNVTV2	000042	CNVTV3	000044	CNVTV4	000045
CN.ERR	004000	CODVER	000023	COMPLT	000176	CONT	000001	CONTCD	152000
CON.ER	007766	CON.ST	000170	COPY4	000000	CPE	000012	CR	000004
CRDY	000010	CRI	000004	CRY	000022	CSERR	000010	CSR	000031
CTMOUT	000170	CYCLE	000552	CYLSTR	170000	C$TYPE	000002	DATEND	033317
DATT	000400	DBAR	000015	DCERR	000001	DCLASS	001000	DCLK	000030
DCMASK	000170	DCN.ST	000001	DCN.TT	000200	DCRD	000004	DCRS	000001
DDC	002000	DDD	004000	DER	000011	DERR	000004	DEVCL	000377
DFAIL	002000	DI	000400	DIAG	000400	DIINTR	001134	DISCON	000204
DISM	000062	DISN	000063	DIVA	005551	DIVB	005554	DIVC	005555
DIVD	005547	DIVDO	005546	DIVX	005550	DIVXIT	005557	DIV512	004127
DMATST	001234	DMBEG	020000	DMBPC	000002	DMODE	040000	DMODT	000012
DMOVH	000011	DMOVL	000010	DMREG0	000007	DMREG1	000001	DMREG2	000002
DMREG3	000003	DMREG4	000004	DMREG5	000005	DMREG6	000006	DMREG7	000000
DMSTR	000013	DMTEMP	000166	DM.BEG	001352	DM.RD	006142	DM.RLP	006144
DM.WR	006430	DOT	006751	DOT1	006752	DOT2	006753	DPF	000026
DPROC	000006	DRAVL	010000	DRDUP	002000	DRDY	000001	DRINIT	004000
DRVCLR	000005	DRVOL	004000	DRV.AT	002000	DRV.AV	004000	DRV.C1	004000
DRV.C2	002000	DRV.C3	001000	DRV.C4	000400	DRV.DB	001000	DRV.DD	004000
DRV.DE	000200	DRV.DF	000020	DRV.DR	000040	DRV.EL	000010	DRV.F0	002000
DRV.DA	000200	DRV.PE	000040	DRV.PS	000002	DRV.RE	000100	DRV.RR	000100
DRV.RU	000001	DRV.SN	170000	DRV.SR	000020	DRV.SU	170000	DRV.S1	010000
DRV.S2	020000	DRV.S3	040000	DRV.S4	100000	DRV.S7	000400	DRV.UM	176000
DRV.U1	010000	DRV.U2	020000	DRV.U3	040000	DRV.U4	100000	DRV.WE	000010
DRV.W1	010000	DRV.W2	020000	DRV.W3	040000	DRV.W4	100000	DSER	000034
DSK.LN	000054	DSLEDS	000236	DSR	000035	DSTSH	000007	DSTSL	000006
DTEMP1	000206	DTMP	000012	DUPVC	001000	DUP.LN	000014	DXFC	010000
D.E.C.	000003	D.STPA	007725	ECC	000007	ECCC	006450	ECCCA	006457
ECCCB	006467	ECCMSK	176000	ECCOA	006471	ECC1	006477	ECC1A	006506
ECC1B	006512	ECC2	006520	ECC2A	006523	ECC9.1	000035	ECC9.2	000046
ECC9.3	000057	ECHOCD	164000	ECMOVE	006672	ECMOVX	006671	ECMOV9	006670
ECOUNT	000134	ECSUMH	022000	ECSUML	000305	EDSEED	000105	EF.BBR	100000
EF.BBU	040000	EF.LOG	020000	ELEV	100000	ELOC	000070	ELPM	000046
ELPN	000035	ELPO	000070	ELPP	000057	ENCACH	033737	ENDCD	131000
EQ	000021	ERECOV	000006	ERET	006730	ERR	100000	ERRBIT	100000
ERRB1	000045	ERRB1A	000026	ERRB1B	000043	ERRB1E	000027	ERRB1L	000034
ERRB1M	000037	ERRB1O	000031	ERRB1Q	000036	ERRB1T	000042	ERRB1Z	000027
ERRB2E	000046	ERREG	000017	ERRINI	040000	ERRINP	100000	ERRIP	040000
ERRRR	006734	ERRRTC	000004	ERRSET	000051	ERRVEC	000377	ERR00	104000
ERR01	000040	ERR02	000100	ERR03	000140	ERR04	000200	ERR05	000240
ERR06	000300	ERR07	000340	ER.BCA	000147	ER.BP1	000003	ER.CID	003334
ER.DMX	000014	ER.HTO	000011	ER.INT	000010	ER.IWR	000017	ER.LOG	003262
ER.MER	000144	ER.MRR	000026	ER.MST	000013	ER.NIM	000012	ER.PRD	000001
ER.PWR	000002	ER.RAP	000004	ER.ROP	000005	ER.RP2	000003	ER.RRD	000006
ER.RTO	000150	ER.RWR	000007	ER.SAP	020000	ER.SOP	030000	ER.SRN	003326
ER.SRP	010000	ER.STP	000146	ER.TMO	000015	ER.UID	003314	ER.VCI	000016
ER.WP2	000003	ETST1	000013	EVAL	000046	E.RPE2	007764	E.WPE2	007760
E311	006535	E335	006555	E345	006575	E350	006577	E900	006675
E910	006702	E915	006715	E917	006716	E920	006717	F	006653
FAILUR	000034	FAIRCT	077400	FCT.MD	000000	FCT.V1	000002	FCT.V2	000003
FDCLK	000020	FDIAG	100000	FLAG	040000	FLOAT	000010	FM.BAD	000001
FM.CNT	000000	FM.DSK	000002	FM.SDI	000003	FORCE	000001	FORICD	025400
FORSCD	046400	FPE	000200	FRERR	000004	FRMCD	177400	FSEEK	000004
FSTST	000153	FTEST	000010	GETID	000253	GETRES	006760	GETSTA	000011
GO	000001	GORD	000001	GOWR	000012	GPRO	010000	GPR1	020000

GPR2	040000	GPR3	100000	GRNGLN	001176	GSEL	001000	GTCCHR	000207
GTCRSP	000170	GTE	000022	GTUCHR	000210	GTURSP	000167	HADD	000017
HANG	000003	HDCOD	170000	HDLMT	000136	HDRTMO	001000	HEADER	000010
HES	100000	HIBYT	177400	HINIB	000360	HIPAR	007774	HI.OFF	000374
HOSTF	000377	HOSTMO	000200	HSTQUE	000204	HU.T3	000136	HU.T4	000136
HU.T5	000136	HU.T6	000136	HU.T7	000145	HU.T9	000152	H11COD	110000
H11XOR	120000	IDENT	000011	IDX	000040	IE	000200	INCRTN	000455
INDIAG	001000	INI	000020	INITM1	005632	INIT1	005631	INPLEN	000007
INPUT	000011	INSEEK	000012	INSTR	000265	INTI	000200	INTR	000010
INTVEC	000164	INVAL	000014	INVPOP	000025	IOACC	020000	IOCLK	010000
IOCNT	002000	IOMSK	000060	IORTY	036000	IORWR	020000	IOSEK	004000
IPINTR	000016	IPREG	000362	IRCI	000002	IUAR	000006	JMPTST	000005
JUMP	000015	LAB	000000	LADD	000007	LATE	000036	LBNMSK	000377
LBNSTR	007400	LBWR	000016	LCOM	000006	LDR2R1	003211	LEDS	170000
LED1	010000	LED2	020000	LED4	040000	LED8	100000	LESS	000002
LFAIL	001000	LF.CON	040000	LF.SNR	000400	LF.SUC	100000	LGADR	036000
LGCKSV	000276	LG.BAD	003341	LG.CNT	003304	LG.DSK	003346	LG.SDI	003275
LM	000064	LN	000240	LNERR	000020	LNPAG	000002	LOBYT	000377
LOGCOD	000100	LOGEND	000342	LOGLEN	000154	LOGPKT	000304	LOG.LN	000034
LONIB	000017	LOPAR	007403	LO.OFF	000003	LRGADR	000272	LSB	000004
LT400	000000	LV1SV1	000212	LV1SV2	000214	LV2CNT	001400	LW	040000
L.BADO	000016	L.BAD1	000017	L.CHVR	000014	L.CNTO	000010	L.CNT1	000011
L.CNT2	000012	L.CNT3	000013	L.CRFO	000002	L.CRF1	000003	L.CSVR	000014
L.EVNT	000007	L.FLGS	000006	L.FMT	000006	L.HDRO	000026	L.HDR1	000027
L.MLUN	000015	L.RTRY	000023	L.SDIO	000030	L.SDI1	000031	L.SDI2	000032
L.SDI3	000033	L.SDI4	000034	L.SDI5	000035	L.SEQ	000005	L.UHVR	000022
L.UNIT	000004	L.UNTO	000016	L.UNT1	000017	L.UNT2	000020	L.UNT3	000021
L.USVR	000022	L.VSEO	000024	L.VSE1	000025	M	000061	MAPENB	100000
MAPFLG	000246	MAPMSC	000077	MAPMSH	077700	MAPMSK	000003	MAPSAV	000244
MAPVAL	100000	MAP.CH	000101	MAP.MO	000103	MAP.M1	000104	MAP.ND	000116
MAP.NX	000102	MAP.OF	000107	MAP.RD	000105	MAP.ST	000115	MAP.S1	000110
MAP.UR	000106	MAP.VO	000111	MAP.V1	000112	MAP.V2	000113	MAP.V3	000114
MAXSTC	000006	MCP.IO	000020	MCP.LN	000030	MCP.RD	000022	MD.ALL	000002
MD.CMP	040000	MD.CSE	020000	MD.ERR	010000	MD.EXC	000040	MD.EXP	100000
MD.FEU	000001	MD.FKC	000001	MD.IMF	000002	MD.NOV	020000	MD.NXU	000001
MD.PRI	000001	MD.RIP	000001	MD.SAV	000004	MD.SCH	004000	MD.SCL	002000
MD.SEC	001000	MD.SEQ	000020	MD.SER	000400	MD.SHD	000020	MD.SPD	000001
MD.SSH	000200	MD.SWP	000004	MD.VOL	000002	MD.WBN	000100	MD.WBV	000040
MD512	126736	MEMSZ	000042	MERR	000014	MINUTE	000074	MLNTST	002257
MLN20	002254	MLN40	002255	MLN44	002256	MODNUM	000440	MP	000100
MRQUE	000146	MSB	000023	MSCPLN	000264	MULBR2	006735	MULBYT	005560
MULR2	006737	MULR2X	006741	MULT	005562	MULTA	005563	MULTB	005564
MUL2RA	006743	MWQUE	000150	M.ABRT	000000	M.ACCH	127400	M.ACCL	000200
M.AVAH	020000	M.AVAL	000043	M.CCDH	127400	M.CCDL	000200	M.CMPH	127400
M.CMPL	000200	M.DAP	000000	M.ERSH	130400	M.ERSL	000360	M.FLUH	121400
M.FLUL	000200	M.GCST	000000	M.GUSH	020000	M.GUSL	000001	M.ONLH	020000
M.ONLL	000047	M.RDH	167400	M.RDL	000200	M.RPLH	120000	M.RPLL	000001
M.SCCL	000001	M.SUCH	020000	M.SUCL	000044	M.WRH	171400	M.WRL	000360
NACLO	000027	NBCB	000200	NBIBAD	000017	NBUFR	000051	NCDONE	000036
NCPE	000032	NCRY	000002	NCSR	000011	NDCLK	000010	NDPF	000006
NDSER	000014	NDSR	000015	NEG	000023	NENDCD	046400	NEQ	000001
NETST1	000033	NFTEST	000030	NLATE	000016	NLSB	000024	NMERR	000034
NMSB	000003	NNEG	000003	NODEID	000017	NOPREG	000000	NOVER	000016
NPKTS	000024	NPOLL	000035	NPRDY	000010	NRPE	000033	NRPOK	000007
NRRDY	000012	NSCAN	000035	NSCB	000010	NSDI	000004	NSECTR	000032
NSEEKS	000156	NSTOP	000031	NTEST	000025	NUPF	000006	NWRC	000017
NZRO	000001	OD	000200	ODDP	000400	ODTAF	000003	ODTCA	000005
ODTCY	000004	OFFMSK	000777	OFFSET	000020	OFFTRK	000002	ONLINE	000213
ONLREC	100000	OPCOD	000377	OPCODE	000144	OPM	000200	OP.ABO	000001
OP.ACC	000020	OP.ACP	000102	OP.ATT	000017	OP.AVA	000100	OP.AVL	000010
OP.CCD	000021	OP.CMP	000040	OP.DAP	000013	OP.DUP	000101	OP.END	000200
OP.ERS	000022	OP.FLU	000023	OP.GCS	000002	OP.GST	000016	OP.GUS	000003

OP.MRD	000030	OP.MWR	000031	OP.ONL	000011	OP.RD	000041	OP.RPL	000024
OP.SCC	000004	OP.SUC	000012	OP.WR	000042	OVER	000036	OVLPER	000072
OW	140000	OWN	100000	P	000002	PABRT	000001	PACTV	040000
PAGESZ	001000	PAR	000002	PARTST	000445	PKIP	000001	PKTBUF	000342
PKTEND	001352	PKTO01	000342	PKTO02	000374	PKTO03	000426	PKTO04	000460
PKTO05	000512	PKTO06	000544	PKTO07	000576	PKTO08	000630	PKTO09	000662
PKTO10	000714	PKTO11	000746	PKTO12	001000	PKTO13	001032	PKTO14	001064
PKTO15	001116	PKTO16	001150	PKTO17	001202	PKTO18	001234	PKTO19	001266
PKTO20	001320	PLOCK	100000	POLL	000015	POLTST	001014	POLY	006744
POLYP	006745	POLYX	006746	PONER	000002	POP1	007424	POP2	007425
PRDY	000030	PSTACK	000076	PSTAT	100000	PTELEN	000002	P.BCNO	000010
P.BCN1	000011	P.BUFH	000013	P.BUFL	000012	P.BUFO	000012	P.BUF1	000013
P.BUF2	000014	P.BUF3	000015	P.BUF4	000016	P.BUF5	000017	P.CMSO	000012
P.CMS1	000013	P.CNTF	000011	P.CNTO	000014	P.CNT1	000015	P.CNT2	000016
P.CNT3	000017	P.CONS	000004	P.CRFO	000002	P.CRF1	000003	P.CSVR	000013
P.CTMO	000012	P.CYL	000026	P.CYLS	000026	P.ELGO	000020	P.ELG1	000021
P.FBBO	000020	P.FBB1	000021	P.FLGS	000006	P.GRP	000025	P.HSTO	000012
P.HST1	000013	P.HTMO	000012	P.LBNO	000020	P.LBN1	000021	P.LCKA	005472
P.LINK	000000	P.LOCK	005473	P.MED1	000020	P.MED2	000021	P.MLUN	000010
P.MOD	000007	P.OPCD	000006	P.RBNS	000031	P.RBNO	000010	P.RBN1	000011
P.RCTC	000031	P.RCTS	000030	P.RFNO	000010	P.RFN1	000011	P.RGIO	000020
P.RGI1	000021	P.RGOO	000022	P.RGO1	000023	P.RSO3	000005	P.RSO6	000010
P.RSO8	000012	P.RS16	000022	P.RS17	000023	P.RS19	000025	P.SHST	000023
P.SHUN	000022	P.STS	000007	P.TRCK	000024	P.UNFL	000011	P.UNIT	000004
P.UNSO	000024	P.UNS1	000025	P.UNTO	000014	P.UNT1	000015	P.UNT2	000016
P.UNT3	000017	P.USVR	000027	P.VCID	000001	P.VRSN	000010	P.VSEO	000026
P.VSE1	000027	Q	000000	QB	001000	QB.RD	005635	QB.WR	006242
QDADM	000001	QREVB	000001	QUESAV	000152	QW	100000	Q.STAT	140000
RAMPE	000100	RAMTST	000505	RAMZAP	000501	RBCAI	000013	RBNCOD	060000
RBNMSK	000077	RBNPRM	050000	RBNSTR	007400	RBNXOR	060000	RBN2ND	030000
RCC	000014	RCI	000003	RCLR	000011	RCMD	000012	RCSRWO	000024
RCSRW1	000025	RCTBLK	000224	RCTOFF	000222	RCT.EN	100000	RCT.MT	000000
RCT.PR	020000	RCT.SE	030000	RCY	000015	RD	000004	RDCMD	000313
RDFST	000004	RDNXT	000014	RDPF	000010	RD.DIA	005640	READ	000001
READCD	013400	RECALB	000216	RECC	000006	RECCT	000007	RELES	000232
REPSTA	000260	RETCNT	007400	REVEND	000133	REVSTR	000123	RGDATH	000272
RGDATL	000271	RIB	000013	RLL	000004	RNGBSH	000162	RNGBSL	000160
RPC	000012	RPE	000013	RPOK	000027	RRDY	000032	RRM	000005
RRSGEN	000001	RSE	000004	RSPCOF	000022	RSPLEN	000020	RSPO	000002
RSPPOF	000023	RSPPTR	000021	RS.BER	000350	RTCS	000005	RTCSO5	000040
RTCSO6	000100	RTCS13	000040	RTCS14	000100	RTDS	000002	RTDSO6	000100
RTOEVE	000010	RT1	000016	RUNN	000014	RUPF	000010	RVACTV	000020
RVCBNH	000131	RVCBNL	000130	RVCBUF	000226	RVCEBF	000230	RVCFLG	000254
RVCIND	000216	RVCRTY	000220	RVCSAV	000123	RVCSDI	000232	RVCT	100000
RVCTA	000132	RVCVEC	000234	RVECDP	000252	RVECUP	000250	RVWRIT	000001
RWM	000015	RWRDY	100000	R0	000000	R1	000001	R10	000010
R11	000011	R12	000012	R13	000013	R14	000014	R15	000015
R16	000016	R17	000017	R2	000002	R3	000003	R4	000004
R5	000005	R6	000006	R7	000007	SAREG	000364	SAVADR	000274
SAVBUF	000270	SAVCNT	000272	SAVEDC	000266	SAVR7	000172	SAVUAR	000262
SCAN	000015	SCLR	000011	SCMD	000002	SC.AOL	000400	SC.BAD	000300
SC.CNT	000140	SC.DCL	000240	SC.DDE	000340	SC.DIS	000400	SC.DSY	000140
SC.DUP	000200	SC.ECC	000340	SC.EC1	000400	SC.EC2	000440	SC.EC3	000500
SC.EC4	000540	SC.EC5	000600	SC.EC6	000640	SC.EC7	000700	SC.EC8	000740
SC.EDC	000100	SC.FER	000000	SC.HDR	000100	SC.IMR	000240	SC.INV	000100
SC.IOP	000100	SC.LVO	000400	SC.NML	000000	SC.NOM	000140	SC.NVL	000040
SC.N12	000240	SC.ODB	000100	SC.ODT	000040	SC.OVR	000040	SC.PAR	000200
SC.POS	000140	SC.RRD	000300	SC.RWR	000200	SC.SCN	000100	SC.SDI	000040
SC.STO	000040	SC.UNK	000000	SC.WPH	020000	SC.WPS	010000	SD	000006
SDIBEG	001355	SDIB.L	000120	SDIRTY	000003	SDIS	000017	SDITMO	000017
SDITO	004400	SDI.AT	000046	SDI.BH	000030	SDI.BL	000027	SDI.CH	000012
SDI.CL	000011	SDI.CP	000002	SDI.CW	000056	SDI.DB	000007	SDI.DL	000010

SDI.DP 000070	SDI.EC 000035	SDI.ER 000035	SDI.ES 000026	SDI.EO 000020
SDI.E1 000017	SDI.FC 000073	SDI.GC 000065	SDI.GO 000074	SDI.GP 000013
SDI.H1 000063	SDI.H2 000064	SDI.I1 000037	SDI.I2 000040	SDI.I3 000041
SDI.LL 000101	SDI.LN 000070	SDI.LT 000034	SDI.L1 000075	SDI.L2 000076
SDI.M1 000071	SDI.M2 000072	SDI.OE 000024	SDI.OM 000025	SDI.OP 000054
SDI.PH 000100	SDI.PL 000055	SDI.PO 000022	SDI.RC 000034	SDI.RH 000032
SDI.RL 000031	SDI.RO 000023	SDI.RS 000036	SDI.RT 000067	SDI.RV 000077
SDI.SD 000057	SDI.SL 000001	SDI.SS 000014	SDI.ST 000000	SDI.SV 000021
SDI.SW 000003	SDI.S1 000044	SDI.S2 000045	SDI.S4 000050	SDI.S5 000051
SDI.S6 000052	SDI.S7 000053	SDI.TG 000066	SDI.TI 000042	SDI.TM 000004
SDI.TO 000033	SDI.UB 000006	SDI.UE 000043	SDI.UF 000060	SDI.UG 000005
SDI.UN 000043	SDI.V1 000061	SDI.V2 000062	SDI.XH 000016	SDI.XL 000015
SDI.XR 000033	SDI.1 001355	SDI.1T 000010	SDI.12 000074	SDI.2 001475
SDI.2T 000047	SDI.3 001615	SDI.4 001735	SDPF 000000	SDTIMO 000200
SEC 000020	SECC 000016	SECCT 000017	SECOND 112000	SECSZ 000400
SECTR 000012	SEEK 000002	SEEKL 000006	SEKPRV 000051	SEKSAV 000052
SENDS2 001110	SENDS3 001112	SENDS4 001114	SEQERR 000003	SEQTST 000017
SEQUEN 000010	SEQ01 000025	SERIAL 000040	SES 040000	SETBAR 006761
SETHVR 007440	SETLIM 001137	SEX2SK 000036	SF 000400	SFERR 000002
SFT134 001253	SFT20 000421	SGRPCD 107000	SHFID 000254	SHFRES 006466
SIMTST 000000	SLAT 020000	SM 000040	SPURT 000040	SRM 000015
SRSGEN 000011	SSE 000014	SSSDIA 001073	SST 002000	STARSP 000366
STBBLG 003366	STBFLG 003365	STBFLH 003360	STCACH 033317	STDALN 000400
STDTBL 007731	STEP 000011	STEP.X 001115	STEP.1 001147	STEP.2 001202
STEP.3 001214	STEP.4 001274	STEP1 004000	STEP2 010000	STEP3 020000
STEP4 040000	STEST 000030	STOP 000011	STOPCM 140000	STOPEN 020000
STRTCD 070400	STROLG 003370	STR1R2 003213	STS 004000	ST.ABO 000002
ST.AVL 000004	ST.CMD 000001	ST.CMP 000007	ST.CNT 000012	ST.DAT 000010
ST.DRV 000013	ST.HST 000011	ST.MFE 000005	ST.MSK 000037	ST.OFL 000003
ST.SHF 000007	ST.SUB 000040	ST.SUC 000000	ST.WPR 000006	SUPF 000000
SUSP 000040	SWM 000005	SY 000112	SYNC 000202	SYNCH 023000
SYNCL 000274	SYO 000112	SY8 000122	SZBUFA 005543	S.ADRH 000023
S.ADRL 000022	S.AXAB 005540	S.BITO 005603	S.BUFR 001104	S.CLRB 005630
S.CL11 005612	S.CYLH 000025	S.CYLL 000024	S.DECB 005614	S.DECO 005615
S.FBC1 005532	S.FMB1 005530	S.FUNB 005534	S.GRUP 000026	S.INC8 005611
S.LBNH 000017	S.LBNL 000016	S.LDCR 005567	S.LDIU 005600	S.LDQO 005572
S.LDRO 005573	S.LDR1 005574	S.LDR2 005575	S.LDR3 005576	S.LDR5 005577
S.LDR6 005600	S.LDR7 005601	S.LDUB 005577	S.LD10 005602	S.LLBO 005604
S.LLB6 005606	S.MLBO 005605	S.MLB6 005607	S.OPFL 000031	S.RELC 005505
S.RELD 005516	S.RELF 005514	S.RLBA 005506	S.RRR1 005501	S.RRR7 005503
S.SDIO 005524	S.SDI1 005522	S.SD11 005526	S.SECI 000031	S.SECS 000030
S.SLB1 005617	S.SSDI 001072	S.STIU 005624	S.STQQ 005613	S.STRO 005616
S.STR1 005620	S.STR2 005621	S.STR3 005622	S.STR5 005623	S.STR6 005624
S.STR7 005625	S.STUB 005623	S.ST10 005626	S.ST11 005627	S.SWB1 005504
S.SXAB 005541	S.SXB1 005536	S.TRAK 000027	S.TST 002240	S.XORO 005610
S.ZBUF 005544	TEMP 000256	TEMP1 000046	TEMP2 000047	TEMP3 000050
TEST 000005	TESTBL 000026	TMR.BS 000176	TMR.MC 000174	TOPOL 000220
TOPRSP 000157	TSBRPE 000467	TSCRPE 000014	TSTCNT 000000	TSTMOD 002263
TSTRTN 000252	TO 000077	T1 000102	T2 000132	T8 000146
T9 000153	UBAR 000005	UBURST 000140	UCRD 000004	UCRS 000014
UCRTST 000456	UCSREN 000400	UC2EDA 002413	UDA1 000001	UDD 000005
UDDI 000010	UDS 000015	UER 000007	UF.CMR 000001	UF.CMW 000002
UF.MSK 000003	UF.RMV 000200	UF.RPL 100000	UF.WPH 020000	UF.WPS 010000
UF.576 000004	UGUCPC 002616	UGUSTA 002625	UGUSTB 002631	UGUSTC 002632
UHANG 000060	UHELP 000055	UIOPA1 002747	UIOPER 003022	UMP 000020
UMPN 000000	UNB.CA 004701	UNB.CB 004720	UNB.CC 004726	UNB.CM 004667
UNB.CX 004427	UNB.RD 005635	UNB.RS 005633	UNB.SR 004704	UNB.WR 006242
UNC.RM 000200	UNC.SS 100000	UNERR 000040	UNITF 177400	UNLOCK 005474
UNSUCC 000175	UN.BRC 005636	UN.BUF 001355	UN.BWC 006243	UPF 000026
UPROC 000026	UP.XFC 007737	UQ.CNT 000000	URAMDA 001625	URAMPE 000475
URETRY 000040	UROMPE 000471	USTRTA 001521	UTEMP1 000210	UTMP 000010
UTSTBL 007426	UWMCI 000006	UXFCRW 005407	UXFC11 005407	UXFC12 005407

UXFC13 005423	UXFC14 005427	UXFC15 005446	UXFC16 005446	UXFC17 005454
UXFC18 005462	UXFUIO 005634	UXF14A 005427	UXF14B 005443	UXF16B 005453
U.ABRT 002274	U.ACC 002670	U.ALOC 003774	U.ALOH 004040	U.ALOL 004062
U.ALOZ 004103	U.ATTA 002336	U.ATTN 002313	U.AVAL 002344	U.AVLA 002362
U.AVLB 002365	U.AVLC 002373	U.AVLO 002347	U.AVL1 002354	U.BDSM 004337
U.BFIL 004161	U.BFLA 004165	U.BFLC 004205	U.BFLF 004217	U.BFLH 004234
U.BFLX 004232	U.BFSV 004147	U.BLST 004471	U.BMTA 004373	U.BMTC 004403
U.BMTD 004415	U.BMTE 004431	U.BMTF 004445	U.BMTI 004460	U.BMTJ 004465
U.BMTK 004362	U.BMTO 004545	U.BMTR 004512	U.BMTS 004553	U.BMTT 004530
U.BMTU 004535	U.BMTV 004542	U.BMTW 004334	U.BMTX 004331	U.BMTY 004327
U.BMTZ 004341	U.BSET 005014	U.BST1 005070	U.CCD 002270	U.CINT 001572
U.CIN1 001574	U.CKEA 005214	U.CKEB 005220	U.CKED 005175	U.CKTC 005224
U.CLGA 004300	U.CLGB 004314	U.CLGC 004315	U.CLGD 004306	U.CLM1 005076
U.CLOG 004273	U.CLRM 005075	U.CLRS 002375	U.CLSX 002376	U.CMDA 002506
U.CMDB 002511	U.CMDC 002513	U.CMDE 002502	U.CMER 002231	U.CMP 003143
U.CMPR 002662	U.CNVT 002430	U.CPGC 004503	U.CPME 005157	U.CPRM 005101
U.CRCT 003215	U.CROS 004655	U.CRSA 004662	U.CSDI 003766	U.CSLA 003336
U.CSPT 002473	U.CSTA 003645	U.CUNK 002477	U.C2ED 002377	U.C2EP 002420
U.C2EZ 002417	U.DAP 002346	U.DATA 003763	U.DERC 004571	U.DERD 004573
U.DERE 004574	U.DERF 004570	U.DERP 004324	U.DIAG 000077	U.DONE 003641
U.DONX 003640	U.END 000215	U.ER 007767	U.ERS 002672	U.EXLP 003763
U.EXSP 003737	U.FLU 002272	U.GCSA 002531	U.GCSB 002533	U.GCSC 002535
U.GCST 002515	U.GETP 005273	U.GLBN 004577	U.GLBX 004602	U.GMST 005476
U.GPKA 002007	U.GPKB 002011	U.GPKT 002005	U.GREF 002536	U.GRFA 002542
U.GRFB 002550	U.GSDA 002553	U.GSDB 002556	U.GSDC 002562	U.GSDD 002566
U.GSDI 002551	U.GSDN 002552	U.GSDW 002572	U.GSDX 002602	U.GSDY 002607
U.GSST 003725	U.GTAD 001777	U.GUCP 002610	U.GUS 002617	U.GUST 002617
U.HERR 002067	U.HTMO 001622	U.HWVR 007750	U.IDLB 001545	U.IDLC 001562
U.IDLD 001564	U.IDLE 001537	U.IMEX 002303	U.INTA 002030	U.INTI 002013
U.INTR 002030	U.INTV 002015	U.INTX 002030	U.INVA 005312	U.INVM 005311
U.IOCM 002757	U.IOPA 002744	U.IOPB 002756	U.IOPC 002770	U.IOPE 003073
U.IOPF 003100	U.IOPG 003110	U.IOPI 003125	U.IOPJ 003131	U.IOPK 003144
U.IOPL 003146	U.IOPM 003176	U.IOPN 003153	U.IOPP 003032	U.IOPR 002705
U.IOPS 002732	U.IOPW 003011	U.IOPX 002761	U.IOPY 002764	U.IOPZ 003034
U.IOPO 002676	U.IOP1 002711	U.IOSA 003172	U.IOSH 003162	U.LNKA 003223
U.LNKB 003224	U.LNKH 003230	U.LNKP 003222	U.LOGA 003245	U.LOGS 003233
U.LRR1 004636	U.L1R1 004136	U.MAP1 005051	U.MAP2 005047	U.MINT 005232
U.MIN1 005265	U.MIOA 003373	U.MIOB 003375	U.MIOP 003372	U.MLUN 003401
U.MNRD 003406	U.MNWR 003413	U.MODL 005477	U.MWDA 003420	U.MWDB 003421
U.NOP 002304	U.ONLA 003436	U.ONLB 003441	U.ONLC 003505	U.ONLD 003520
U.ONLE 003527	U.ONLF 003546	U.ONLG 003573	U.ONLH 003600	U.ONLI 003607
U.ONLL 003631	U.ONLN 003424	U.ONLR 003563	U.ONLS 003504	U.ONLX 003554
U.ONLY 003564	U.ONLZ 003500	U.ONLO 003426	U.OPCD 002166	U.OPCE 002223
U.OPDM 002224	U.OPIM 002215	U.OVLA 004643	U.OVLB 004644	U.OVLP 004637
U.PKTZ 003206	U.PTEL 005314	U.PTL2 005341	U.PTL3 005323	U.PTL4 005326
U.PTL5 005345	U.QSND 003340	U.RAMD 001624	U.RAMX 001541	U.RCSA 001745
U.RCSB 001746	U.RCSC 001747	U.RCSR 001737	U.RCVE 001735	U.RD 002664
U.RECV 001675	U.RPL 002674	U.RRCA 004732	U.RRCC 004754	U.RRCE 004767
U.RRCX 004741	U.RREC 004731	U.RSTR 000002	U.RSVA 002241	U.RSVB 002247
U.RTST 002233	U.RVCK 004323	U.RVCT 004137	U.RVC1 006762	U.RVC3 006762
U.RVC5 007022	U.RVC7 007153	U.RVFA 007144	U.RVFB 007141	U.RVFE 007114
U.RVFH 007140	U.RVRL 007147	U.RVRT 007151	U.RVUX 007020	U.RV3A 006771
U.RV3B 007015	U.RV3C 006774	U.RV3D 007016	U.RV5A 007033	U.RV5B 007066
U.RV5C 007076	U.RV5D 007101	U.RV5E 007103	U.RV5F 007111	U.RV5G 007126
U.RV5J 007031	U.RV5R 007053	U.RV5T 007054	U.RV5W 007047	U.RV5X 007041
U.RV5Y 007046	U.RV5Z 007061	U.RV57 007022	U.RV7A 007153	U.R1R0 005223
U.R1R1 005310	U.SABT 003766	U.SADQ 002004	U.SADX 002003	U.SBCN 003210
U.SCCH 003653	U.SCID 007751	U.SEKA 002077	U.SEKB 002107	U.SEKC 002112
U.SEKG 002147	U.SEKH 002150	U.SEKI 002103	U.SEKO 002101	U.SEND 001627
U.SETS 003102	U.SETX 003103	U.SGRP 004664	U.SLBN 004604	U.SLEN 003337
U.SNDA 001647	U.SNDB 001664	U.STER 007733	U.STOF 001753	U.STRT 001500
U.STSX 003107	U.STUD 003721	U.ST1U 002472	U.SUCH 003422	U.SUID 003675

LSCS FORM=QUAD

```
U.SUNI 003702    U.TBEC 004624    U.TIMR 001603    U.TMDA 004320    U.TMOD 004321
U.TMRA 001610    U.UCSR 001755    U.UERD 002066    U.UERR 002071    U.ULKA 003711
U.ULNK 003713    U.UNEA 004255    U.UNEB 004266    U.UNEC 004252    U.UNED 004241
U.UNEE 004246    U.UNER 004247    U.UNRD 004211    U.UNWR 004407    U.VAXP 002014
U.VAXR 002027    U.VCER 002176    U.WR   002666    U.WTSA 004627    U.WTST 004625
U.XCMA 004472    U.XCMP 005230    U.XFCP 005402    U.ZPSW 003371    U8K    000100
VAXINT 000400    VAXO   000006    VAX.PG 000400    VC.CMD 000000    VC.DMM 177760
VC.LOG 000020    VC.RSP 000000    VECT   000010    VECTAB 007725    V.RVCT 007725
V.RVC3 007726    V.RVC5 007727    V.RVC7 007730    V.RVC8 007731    V.URVC 007725
WAIT   000243    WAITS  000246    WASTE  000275    WCI    000005    WD     000000
WMCI   000007    WRAP   000000    WRC    000037    WRCMD  000302    WRFST  000005
WRITCD 122400    WRITE  000004    WRNXT  000015    WRPFLG 100000    WRRDXR 131000
WRRDX1 175377    WRSND  000003    WRT    000002    WRTERR 000366    WRTSA  000365
WTCMDD 000323    WT.DIA 006245    WT.RAM 001077    XBNCOD 120000    XBNSTR 170000
XCMP   000200    XFCFIN 000021    XFCMAX 000022    XFCTAB 007737    XFCUPR 000012
XWRCMD 002050    XWTCMD 002061    ZRO    000021    $DIG1  000110    $DIG2  000113
$DIG3  000115    $DMA1  001241    $DMA2  001245    $END   010000    $HVEC  007760
$PAR   000445    $SETL  001143    $SVEC  007400    $$BDA  000002    $$KDA  000001
$$KDB  000002    $$QDA  000001    $$UDA  000000
```

```
KDBUP            DIGITAL EQUIPMENT CORP., 2901 ASSEMBLER VERSION 32 PAGE S-1
CROSS REFERENCE TABLE (CREF V01-08 )

$$BDA     1-0       8-0      12-0      13-0      14-0      14-0      15-0      15-0      18-0      22-0      30-0      38-0      45-0      46-0
         51-0      51-0      51-0      51-0      64-0      65-0      67-0      71-0      73-0      74-0      74-0      76-0      76-3      76-3
         77-11     77-15     79-26     80-37     80-54     80-55     82-75     83-111    84-116    84-122    85-142    85-153    91-252    91-253
        102-456   102-456   102-456   102-460   106-530   126-1115  128-1137  129-1147  132-1210  133-1215  133-1220  133-1222  138-1300  139-1322
        139-1325  140-1515  142-1541  142-1541  145-1604  148-1701  148-1704  152-2002  153-2004  155-2014  155-2023  155-2023  160-2072  191-2730
        193-3007  202-3344  212-3614  220-3725  221-3743  232-4247  232-4254  237-4510  240-4627  246-4771  248-5022  249-5057  249-5075  251-5127
        251-5136  259-5352  278-5541  278-5542  286-5633  289-5634  297-6201  301-6242  329-7400  329-7404  331-7440  332-7445  333-7445  334-7750
        334-7760
$$KDA     1-0       1-0      268-5466
$$KDB     1-0       1-0      268-5466
$$QDA     1-0       1-0       8-0      12-0      13-0      15-0      15-0      18-0      30-0      45-0      46-0      63-0      65-0      65-0
         65-0      65-0      71-0      76-0      76-3      77-11     77-15     77-16     80-45     80-55     82-75     83-112    84-125    85-153
        102-456   102-456   102-466   122-1072  124-1075  126-1110  126-1115  128-1137  128-1145  129-1147  133-1222  138-1300  139-1315  139-1321
        140-1514  148-1706  160-2072  176-2424  191-2725  196-3136  203-3355  206-3410  207-3414  212-3612  220-3725  221-3741  222-3763  230-4212
        235-4410  243-4706  248-5016  248-5034  249-5065  249-5075  253-5201  259-5333  259-5342  278-5542  287-5634  287-5634  288-5634  288-5634
        288-5634  296-6170  297-6201  300-6242  300-6242  300-6242  329-7404  330-7426  330-7426  333-7734  334-7750
$$UDA     1-0       8-0      12-0      13-0      15-0      15-0      18-0      20-0      21-0      22-0      22-0      25-0      27-0      29-0
         45-0      46-0      59-0      67-0      70-0      74-0      74-0      75-0      76-0      76-3      77-11     77-15     79-27     79-31
         80-55     82-75     82-75     83-105    84-114    84-130    85-147    85-153   103-475   104-500   124-1077  124-1104  126-1115  128-1137
        128-1146  129-1147  133-1216  133-1222  134-1233  134-1234  143-1572  145-1624  160-2072  201-3305  220-3725  226-4070  230-4206  230-4211
        232-4241  235-4404  235-4407  237-4503  238-4512  243-4704  248-5014  249-5075  253-5177  256-5232  278-5536  278-5542  282-5567  285-5633
        306-6450  334-7750
$DIG1     83-110
$DIG2     83-113
$DIG3     84-115
$DMA1    136-1241
$DMA2    136-1245
$END     335-8000
$HVEC    334-7760
$PAR     101-445
$SETL    128-1143
$SVEC    329-7400
A        306-6450  315-6604  315-6621  317-6667
ACLO      12-0      13-0      13-0
ALGADR    65-0      65-0      65-0     137-1263  238-4514  238-4521  315-6606  319-6743
ALGERR   238-4512  238-4512
ALOLMT    48-0      48-0     155-2015
ALUTST    83-77     99-373
ATTCOD    45-0     200-3243
ATTN      25-0
AVAIL     25-0
AVL.LN    52-0     201-3261
B        306-6450  315-6611  315-6617
B.BAD     30-0      76-1      76-3      80-55     80-56     81-73     95-335    98-366   160-2072  160-2075
B.BTO     42-0
B.CPE     42-0     115-715
B.CTE     42-0
B.DTYP    42-0      87-203    87-204
B.ICE     42-0
B.ISE     42-0
B.IVE     42-0
B.LED     30-0      86-154    86-170    98-365   122-1070  122-1071  126-1120  129-1153  130-1162  133-1221  133-1223  133-1224  139-1323  148-1701
        156-2035  156-2036  289-5643  299-6212  299-6213  299-6237  299-6240  301-6250
B.LOOP    30-0      76-1      76-3      80-55     80-56     81-73     86-154    95-335    98-365    98-366   122-1070  160-2072  160-2075
```

LSCS FORM=QUAD

```
B.MPE    42-0
B.MTCE   42-0
B.NABO   30-0      80-56     86-154    86-155    86-170    98-365    98-366   107-531   108-552   115-712   116-742   117-754   119-1022  122-1070
        122-1071  126-1120  129-1153  130-1162  133-1221  133-1224  139-1323  148-1702  156-2036  289-5643  299-6213  299-6240  301-6250  334-7771
        335-7772
B.NEX    42-0
B.NMR    42-0
B.NRTY   30-0      76-1      76-3      80-55     80-56     81-73     86-154    86-170    95-335    98-365    98-366   122-1070  122-1071  126-1120
        129-1153  130-1162  133-1221  133-1223  133-1224  139-1323  148-1701  156-2035  156-2036  160-2072  160-2075  289-5643  299-6212  299-6213
        299-6237  299-6240  301-6250
B.ODD    30-0
B.RDS    42-0
B.RTO    42-0
B.SPE    42-0     116-745
B.STO    42-0
B.TDF    42-0
BA       21-0      30-0      43-0     128-1137  128-1143  278-5541
BA16     21-0      30-0      43-0      43-0
BA17     21-0      30-0      43-0      43-0
BA18     30-0      42-0      43-0
BA19     30-0      42-0      43-0
BA20     30-0      42-0      43-0
BA21     30-0      42-0      43-0
BA22     42-0      43-0
BA23     42-0      43-0
BA24     42-0      43-0
BA25     42-0      43-0
BA26     42-0      43-0
BA27     42-0      43-0
BA28     42-0      43-0
BA29     42-0      43-0
BAD.LN   52-0     202-3345
BADRH    51-0      86-160    92-270    98-357
BADRL    51-0      91-263    92-265    98-354
BANOT   191-2730  193-3007  202-3344  251-5137  259-5352  278-5541  278-5541
BAR      18-0      76-0      79-32     84-132    86-160    87-201    91-263    92-265    92-270    93-305    93-307    93-311    94-317    98-354
        98-357    99-377    104-475   104-501   106-512   106-515   106-527   107-530   108-545   108-556   108-560   108-563   108-565   108-570
       108-572   108-576   108-603   109-606   110-611   110-613   110-615   110-617   110-623   110-625   114-643   115-705   115-707   116-727
       116-731   116-733   117-751   117-762   117-770   117-772   118-1002  118-1004  118-1010  118-1012  122-1051  122-1052  122-1057  122-1064
       122-1066  124-1073  124-1101  125-1105  125-1107  126-1132  128-1146  131-1167  131-1171  132-1203  132-1206  132-1210  132-1212  133-1215
       133-1216  134-1226  134-1227  134-1231  134-1233  136-1235  136-1242  136-1247  136-1251  137-1263  137-1264  137-1273  138-1276  139-1320
       140-1500  140-1501  140-1503  140-1505  140-1510  140-1512  140-1515  140-1522  140-1526  140-1530  140-1531  140-1533  140-1535  142-1537
       142-1540  142-1544  142-1547  142-1553  142-1554  142-1556  142-1561  143-1562  143-1565  143-1570  144-1574  145-1610  145-1614  145-1622
       145-1623  145-1624  145-1626  146-1636  146-1640  146-1641  147-1646  147-1647  147-1651  147-1654  147-1655  147-1656  147-1657  147-1660
       147-1664  147-1671  149-1712  149-1715  149-1716  149-1721  149-1724  149-1725  149-1727  149-1732  149-1734  149-1736  150-1737  150-1740
       150-1742  150-1744  150-1746  150-1751  151-1755  151-1756  151-1762  151-1764  151-1772  151-1775  152-1777  152-2000  154-2010  154-2011
       155-2015  155-2022  158-2053  158-2055  158-2057  160-2071  161-2077  161-2101  161-2103  162-2112  162-2113  162-2116  162-2120  162-2122
       162-2123  162-2124  162-2127  162-2131  162-2145  162-2150  163-2151  163-2152  163-2154  163-2155  163-2156  163-2160  163-2163  164-2166
       164-2167  164-2170  165-2225  165-2231  166-2233  167-2242  167-2252  168-2257  169-2264  169-2266  172-2317  172-2322  172-2325  172-2326
       173-2331  173-2333  173-2334  173-2342  174-2354  174-2362  174-2365  174-2366  175-2370  176-2405  176-2413  176-2420  176-2421  177-2426
       179-2430  179-2431  179-2432  179-2433  179-2437  179-2442  179-2443  179-2447  179-2452  179-2456  179-2460  179-2461  180-2467  180-2471
       180-2472  180-2473  180-2474  181-2477  182-2503  182-2504  182-2507  182-2511  182-2512  183-2523  183-2525  183-2531  183-2532  183-2533
       184-2536  184-2537  184-2542  184-2544  184-2545  184-2547  185-2551  186-2573  186-2577  187-2610  187-2611  188-2622  188-2623  188-2631
       188-2632  188-2633  188-2634  188-2635  189-2645  189-2646  189-2647  189-2650  189-2652  189-2653  189-2655  191-2720  191-2725  191-2727
```



```
       192-2732  192-2733  192-2740  192-2756  192-2757  192-2760  192-2761  192-2762  193-2772  193-2775  193-3000  193-3002  193-3005  193-3010
       193-3011  193-3012  193-3013  193-3014  193-3020  193-3022  193-3024  193-3025  193-3027  193-3032  193-3033  194-3034  194-3035  194-3037
       194-3041  194-3045  194-3051  194-3052  194-3056  194-3057  194-3062  195-3066  195-3067  195-3074  195-3075  195-3076  195-3100  195-3103
       195-3106  195-3111  195-3112  195-3113  196-3116  196-3125  196-3130  196-3135  196-3136  197-3146  197-3153  197-3154  197-3155  197-3156
       197-3165  197-3173  197-3175  197-3202  197-3204  198-3210  198-3212  198-3214  198-3215  198-3216  199-3223  199-3226  199-3230  199-3232
       200-3233  200-3235  200-3236  200-3237  200-3240  200-3241  200-3242  200-3247  200-3251  201-3254  201-3257  201-3265  201-3266  201-3267
       201-3275  201-3276  201-3277  201-3300  201-3301  201-3302  201-3304  201-3312  202-3315  202-3320  202-3321  202-3322  202-3324  202-3330
       202-3331  202-3332  202-3337  202-3340  202-3342  202-3343  203-3347  203-3351  203-3353  203-3355  203-3356  203-3370  203-3371  204-3372
       204-3375  204-3376  204-3377  208-3440  209-3442  209-3443  209-3445  209-3457  209-3460  209-3473  209-3500  210-3506  210-3512  210-3522
       210-3523  210-3524  210-3527  210-3530  210-3531  210-3534  210-3535  210-3536  210-3537  211-3542  211-3543  211-3544  211-3546  211-3554
       211-3555  211-3560  211-3564  211-3565  211-3573  211-3574  211-3603  211-3604  212-3611  212-3612  212-3613  212-3614  212-3616  212-3622
       212-3624  212-3627  212-3630  212-3632  212-3633  212-3634  212-3635  212-3636  212-3637  213-3640  213-3644  213-3645  213-3652  214-3657
       214-3661  214-3662  214-3663  214-3670  214-3672  216-3675  216-3676  216-3677  216-3700  217-3703  217-3705  217-3706  218-3712  218-3714
       218-3720  220-3734  220-3735  221-3752  221-3754  221-3760  221-3761  223-3773  224-3776  224-4001  225-4007  225-4010  225-4011  225-4015
       225-4016  225-4020  225-4021  225-4023  225-4027  225-4030  225-4033  225-4034  225-4037  225-4042  225-4043  225-4044  225-4046  225-4047
       226-4052  226-4053  226-4056  226-4061  226-4062  226-4063  226-4064  226-4066  226-4070  226-4071  226-4072  226-4077  226-4103  226-4104
       227-4110  227-4112  227-4113  227-4115  227-4116  227-4117  227-4120  227-4122  227-4124  227-4125  228-4137  228-4140  228-4142  228-4145
       229-4147  229-4151  229-4154  229-4155  229-4161  229-4165  230-4167  230-4171  230-4175  230-4201  230-4203  230-4212  231-4222  231-4225
       231-4226  231-4232  231-4237  232-4243  232-4250  232-4255  232-4263  232-4266  232-4272  233-4276  233-4300  233-4307  233-4310  233-4311
       233-4312  233-4314  233-4320  233-4321  233-4323  233-4324  234-4331  234-4332  234-4335  234-4342  234-4345  234-4356  234-4360  234-4365
       234-4366  235-4367  235-4373  235-4375  235-4402  235-4410  235-4417  235-4420  236-4427  236-4431  236-4432  236-4434  236-4440  236-4444
       236-4445  236-4446  237-4452  237-4453  237-4460  237-4465  237-4467  237-4473  237-4505  237-4506  238-4514  238-4515  238-4521  238-4522
       238-4525  238-4526  238-4527  238-4530  238-4533  238-4535  238-4540  239-4547  239-4550  239-4551  239-4553  239-4554  239-4556  239-4570
       239-4571  239-4572  240-4577  240-4600  240-4602  240-4604  240-4605  240-4614  240-4615  240-4623  240-4624  240-4625  241-4632  241-4633
       241-4640  241-4645  241-4647  241-4652  241-4653  242-4656  242-4661  242-4662  242-4663  242-4664  242-4665  242-4666  243-4672  243-4677
       243-4701  243-4703  243-4706  244-4710  244-4713  244-4714  244-4716  244-4717  244-4720  244-4721  244-4727  245-4731  245-4732  245-4736
       245-4743  245-4744  246-4747  246-4755  246-4765  246-4767  246-4771  246-4773  246-4774  246-4776  247-5004  248-5014  248-5015  248-5016
       248-5021  248-5022  248-5023  248-5024  248-5027  248-5031  248-5033  248-5034  248-5035  248-5037  248-5040  249-5042  249-5043  249-5044
       249-5045  249-5047  249-5051  249-5052  249-5063  249-5064  249-5065  249-5070  249-5071  249-5072  249-5073  249-5074  249-5075  249-5077
       250-5101  250-5102  250-5104  250-5107  250-5111  250-5112  250-5115  250-5116  250-5117  250-5121  251-5124  251-5125  251-5126  251-5127
       251-5132  251-5135  251-5136  251-5140  251-5141  251-5143  251-5146  251-5147  251-5150  251-5151  251-5152  251-5154  251-5156  251-5157
       251-5160  252-5165  252-5166  252-5171  252-5173  253-5175  253-5176  253-5200  253-5201  253-5211  253-5212  253-5213  253-5216  253-5221
       255-5224  255-5225  256-5230  257-5233  257-5234  257-5235  257-5236  257-5237  257-5240  257-5242  257-5243  257-5244  257-5245  257-5246
       257-5247  257-5250  257-5251  257-5252  257-5253  257-5254  257-5256  257-5257  257-5265  257-5267  257-5270  258-5273  258-5274  258-5275
       258-5314  259-5315  259-5320  259-5321  259-5327  259-5330  259-5334  259-5337  259-5341  259-5343  259-5345  259-5351  259-5354  259-5360
       259-5361  259-5364  259-5365  259-5366  259-5372  259-5373  259-5374  259-5375  259-5376  259-5400  262-5407  262-5410  262-5411  262-5414
       262-5416  262-5421  263-5424  263-5425  264-5430  264-5432  264-5435  264-5436  264-5444  266-5446  266-5450  267-5454  268-5462  268-5465
       268-5466  270-5475  273-5505  273-5507  273-5511  273-5517  278-5542  279-5544  282-5570  289-5636  289-5640  289-5642  290-5673  290-5700
       290-5707  290-5714  290-5716  290-5724  290-5731  290-5733  291-5741  291-5746  291-5750  291-5756  291-5763  291-5765  291-5773  291-6000
       291-6002  292-6010  292-6015  292-6017  292-6025  292-6032  292-6034  293-6043  293-6044  293-6045  293-6046  293-6055  293-6056  293-6057
       293-6063  293-6103  294-6107  295-6125  295-6133  296-6147  296-6156  296-6157  296-6160  296-6161  296-6162  296-6163  296-6164  299-6231
       299-6232  299-6233  299-6235  301-6243  301-6245  301-6247  301-6263  301-6273  301-6276  302-6302  302-6312  302-6314  302-6316  302-6320
       302-6322  302-6324  303-6335  303-6337  303-6341  303-6353  303-6355  304-6375  304-6377  304-6401  304-6413  304-6415  306-6432  306-6437
       307-6450  307-6451  307-6452  308-6472  308-6474  309-6477  309-6501  309-6507  309-6513  309-6514  309-6515  309-6516  311-6523  311-6524
       311-6525  311-6530  311-6531  312-6533  312-6534  312-6535  312-6542  312-6543  312-6546  312-6550  312-6555  312-6556  313-6562  313-6564
       313-6571  313-6572  313-6573  313-6574  315-6601  315-6604  315-6606  315-6610  315-6612  315-6613  315-6615  315-6622  315-6624  315-6625
       315-6626  316-6636  316-6640  316-6644  316-6645  317-6647  317-6650  317-6651  317-6660  317-6664  317-6672  317-6673  318-6675  318-6676
       318-6701  318-6702  318-6713  318-6722  318-6724  318-6730  318-6731  318-6732  318-6736  319-6743  319-6746  319-6747  319-6753  319-6754
       319-6756  319-6760  319-6761  322-6762  322-6764  322-6765  322-6766  322-6770  322-6773  322-6774  322-6775  322-6776  322-7000  322-7001
       322-7002  322-7004  322-7005  322-7007  322-7010  322-7012  322-7016  322-7020  323-7022  323-7023  323-7033  323-7035  323-7041  323-7046
       323-7050  323-7053  323-7054  323-7055  323-7057  324-7062  324-7063  324-7064  324-7070  324-7074  324-7101  324-7103  324-7105  324-7107
       324-7111  324-7114  324-7120  324-7122  324-7124  324-7127  325-7135  325-7136  326-7140  326-7142  326-7144  326-7147  326-7150  326-7151
       326-7152  327-7154  327-7156  327-7157  327-7162  333-7734  333-7735  333-7741  333-7742  333-7743  334-7750
BCAID    18-0      81-64     81-65     81-66     92-267    92-271    92-272    92-274    93-304    93-310    93-312    94-315    98-370    98-371
```

```
        108-541  108-543  108-555  108-557  108-564  108-571  108-573  113-634  113-635  113-636  114-644  114-660  115-676  116-726
        116-732  116-734  117-767  117-771  117-773  118-1000 158-2052 158-2056 158-2060 289-5654 289-5655 289-5657 293-6076 293-6100
        293-6102 295-6116 295-6121 295-6124 296-6144 296-6145 296-6146 297-6174 297-6200 299-6221 301-6261 301-6262 301-6264 301-6271
        301-6272 301-6274 301-6277 301-6300 302-6313 302-6315 302-6317 302-6321 302-6323 303-6332 303-6333 303-6334 303-6336 303-6340
        303-6347 303-6350 303-6352 303-6354 304-6363 304-6372 304-6373 304-6374 304-6376 304-6400 304-6410 304-6412 304-6414 305-6424
        306-6430 306-6431 306-6434 306-6435 306-6436 306-6440
BCAIS    18-0     94-321   94-322  108-601  109-604  110-610  110-612  110-614  110-616  110-620  114-647  115-706  115-710  118-1003
        118-1005 290-5674 290-5675 290-5701 290-5702 290-5710 290-5711 290-5715 290-5717 290-5725 290-5726 290-5732 291-5734 291-5742
        291-5743 291-5747 291-5751 291-5757 291-5760 291-5764 291-5766 291-5774 291-5775 291-6001 291-6003 292-6011 292-6012 292-6016
        292-6020 292-6026 292-6027 292-6033 292-6035 293-6042 293-6047 293-6050 293-6051 293-6053 293-6054 293-6060 293-6067 293-6070
        293-6071 293-6072 293-6073 293-6074 294-6106 295-6132 296-6155 299-6224 299-6225
BCATST   86-162  107-530
BCGRP    64-0    246-4750
BCICSR   41-0    119-1014 119-1017
BDCST    40-0
BDSNF    64-0    234-4334 234-4346 236-4435 247-5006
BECC     64-0    211-3570 234-4334 234-4344 247-5006 323-7024
BECER    64-0    234-4361 235-4370 235-4415 243-4687 247-5006
BERDN    64-0    246-4751 247-5008
BERMAX   67-0     67-0    201-3306
BFRQ     59-0    195-3065 195-3070 195-3073 223-3772 224-3775 225-4050 237-4471 237-4476 237-4477 238-4532 238-4534 241-4634 241-4641
BFSV     59-0    163-2161 163-2164 211-3563 213-3651 223-3771 229-4152 231-4233 232-4257 234-4340 237-4500 247-5013
BFULL    64-0    229-4156 229-4162 231-4231 231-4236 234-4333 237-4455 237-4456 237-4457 237-4464 241-4643 246-5003 327-7155
BGOOD    64-0    247-5006
BGRUP    64-0    230-4204 246-4753 246-5001 247-5005 251-5134 323-7025
BI.ERR  294-6105 295-6127 296-6154 299-6210 302-6311 303-6326 303-6344 303-6357 304-6366 304-6402 306-6444
BI.EX    293-6075 294-6112 295-6141 299-6231 303-6327 303-6360 306-6446
BI.RD    289-5635
BI.RLP   289-5643 295-6140
BI.STO   148-1701 289-5643 299-6212 299-6235 301-6250
BI.STP    86-155   93-311   94-315  107-531  108-551  117-754  117-772  118-1000 119-1022 133-1223 156-2034 158-2057 299-6221 299-6237
        299-6241
BI.TO    290-5676 290-5703 290-5712 290-5720 290-5727 291-5735 291-5744 291-5752 291-5761 291-5767 291-5776 291-6004 292-6013 292-6021
        292-6030 292-6036 293-6061 293-6065 294-6104 295-6126 296-6153 298-6201 298-6207 302-6301 303-6325 303-6341 303-6356 304-6401
        304-6416 306-6443
BI.UDO   293-6066 302-6303
BI.UDX   290-5672 290-5677 290-5705 290-5706 290-5713 290-5722 290-5723 290-5730 291-5737 291-5740 291-5745 291-5754 291-5755 291-5762
        291-5771 291-5772 291-5777 292-6006 292-6007 292-6014 292-6023 292-6024 292-6031 292-6040 293-6062 293-6065
BI.WOW   304-6363 304-6407 305-6424
BI.WR    289-5634 301-6242
BIBAD    13-0
BIBER    41-0     97-352  115-714  116-744  299-6217
BICSR    41-0     86-163   87-206   87-210   91-253  120-1032
BICSRE   43-0    119-1016
BIDTYP   41-0     87-177   87-205
BIEADR   41-0    120-1042
BIECSR   41-0
BIGPRO   41-0
BIGPR1   41-0
BIGPR2   41-0    112-630  114-654  115-672  116-720  117-765  118-776
BIGPR3   41-0
BIIDES   41-0
BIIMSK   41-0
BIITST   86-167   86-172  117-751
```



```
BIMEM    86-175  120-1026 120-1026
BIPE     86-166  112-630
BIPSDE   41-0
BIPSFC   41-0
BIPSRC   41-0
BIR.1W  290-5733 292-6016 293-6047 293-6057 293-6074 296-6156 296-6157 296-6160 296-6161 296-6162 296-6163 296-6164
BIR.2W  291-5750 291-6001 293-6043 293-6044 293-6045 293-6050 293-6055 293-6056 293-6073
BIR.3W  291-5764 291-5765 293-6072
BIR.4W  291-5747 291-6002 293-6071
BIR.5W  290-5732 292-6017 293-6070
BIR.6W  290-5674 290-5701 290-5710 290-5715 290-5725 291-5742 291-5757 291-5774 292-6011 292-6026 292-6034 293-6067
BIR.CM  293-6046 293-6051 293-6060
BIR.EV  293-6041 293-6053
BIR.EX  293-6064 293-6075 296-6167
BIR.L2  290-5673 290-5700 290-5707 290-5714 290-5724 290-5731 291-5741 291-5746 291-5756 291-5763 291-5773 291-6000 292-6010 292-6015
        292-6025 292-6032 293-6052
BIR.LP  293-6041 293-6063
BIR.LW  289-5670 289-5671 293-6061
BIR.NI  293-6042 293-6047 293-6053
BIR.NO  289-5647 295-6113
BIR.OD  293-6042 293-6052
BIR.QK  293-6060 293-6062
BIR.SO  289-5650 295-6114
BIR.X2  293-6076 295-6113
BIRD00  290-5703 329-7404
BIRD01  290-5720 329-7405
BIRD02  291-5735 329-7406
BIRD03  291-5752 329-7407
BIRD04  291-5767 329-7410
BIRD05  291-6004 329-7411
BIRD06  292-6021 329-7412
BIRD07  292-6036 329-7413
BIRD08  290-5676 329-7414
BIRD09  290-5712 329-7415
BIRD10  290-5727 329-7416
BIRD11  291-5744 329-7417
BIRD12  291-5761 329-7420
BIRD13  291-5776 329-7421
BIRD14  292-6013 329-7422
BIRD15  292-6030 329-7423
BIRDBR  289-5665 329-7404
BIRDCM   40-0     94-314   95-334  114-657  115-675  118-777  289-5657 293-6101 295-6123 296-6142 299-6220
BIREOK  289-5645 294-6112 301-6252
BIRTRY   51-0
BISADR   41-0    120-1037
BITO0    18-0     19-0     24-0     25-0     30-0     36-0     39-0     45-0     53-0     58-0     58-0     58-0     58-0     58-0
         59-0     59-0     59-0     60-0     61-0     63-0     64-0     65-0     69-0     69-0     91-260   91-261   99-374  107-534
        107-535  108-554  108-574  114-645  114-653  132-1211 132-1213 163-2164 164-2171 164-2171 164-2171 164-2171 175-2367 175-2371
        176-2406 188-2623 202-3326 202-3326 202-3326 202-3326 209-3474 230-4204 231-4224 233-4316 259-5362
BITO1    18-0     24-0     25-0     39-0     45-0     53-0     58-0     58-0     59-0     59-0     60-0     61-0     63-0     64-0
         70-0    114-656  114-665  115-674  115-711  116-722  116-741
BITO2    18-0     24-0     39-0     58-0     58-0     59-0     60-0     63-0     64-0     69-0
BITO3    18-0     24-0     30-0     39-0     43-0     59-0     61-0     62-0     64-0     69-0
BITO4    18-0     22-0     24-0     25-0     30-0     39-0     49-0     58-0     58-0     59-0     60-0     61-0     62-0     64-0
```

LSCS FORM=QUAD

```
        70-0
BIT05   18-0    22-0    24-0    24-0    25-0    30-0    39-0    58-0    59-0    59-0    60-0    61-0    62-0    64-0
        70-0    73-0    229-4147
BIT06   18-0    21-0    22-0    24-0    24-0    25-0    25-0    30-0    39-0    49-0    58-0    58-0    59-0    61-0
        62-0    73-0    268-5470 268-5471
BIT07   18-0    22-0    24-0    30-0    35-0    35-0    39-0    43-0    50-0    58-0    58-0    59-0    61-0    62-0
        63-0    63-0    69-0    73-0    131-1200 258-5276
BIT08   18-0    21-0    22-0    39-0    43-0    46-0    50-0    58-0    59-0    62-0    62-0    64-0    73-0    73-0
        73-0    102-456 183-2527 196-3145 239-4552 239-4555 239-4557
BIT09   18-0    19-0    39-0    50-0    58-0    59-0    62-0    62-0    64-0    73-0    73-0
BIT10   18-0    21-0    22-0    39-0    42-0    58-0    59-0    60-0    62-0    62-0    63-0    64-0    80-51   137-1266
        137-1270 268-5471 323-7042
BIT11   18-0    19-0    21-0    22-0    35-0    39-0    42-0    45-0    58-0    59-0    60-0    62-0    62-0    62-0
        64-0
BIT12   18-0    19-0    22-0    35-0    39-0    42-0    56-0    58-0    59-0    60-0    61-0    62-0    62-0    63-0
        64-0    139-1325 161-2107 329-7400 333-7445
BIT13   18-0    19-0    22-0    35-0    39-0    43-0    56-0    58-0    58-0    59-0    59-0    60-0    61-0    62-0
        62-0    63-0    64-0
BIT14   18-0    19-0    22-0    35-0    35-0    39-0    42-0    43-0    53-0    58-0    59-0    59-0    60-0    61-0
        62-0    62-0    64-0    64-0    106-521 116-724 129-1152
BIT15   18-0    19-0    22-0    25-0    35-0    35-0    39-0    42-0    43-0    45-0    45-0    45-0    53-0    58-0
        59-0    59-0    60-0    61-0    61-0    61-0    61-0    62-0    62-0    63-0    64-0    64-0    73-0    98-363
        99-373  99-407  99-417  106-521 116-724 139-1325 146-1630 147-1662 148-1704 149-1723 161-2102 162-2114 162-2132 183-2524
        184-2544 192-2745 194-3047 194-3047 203-3361 211-3575 224-4005 226-4072 236-4447 243-4700 246-4745 246-4777 247-5005 248-5023
        258-5273 259-5323 259-5324 259-5350 269-5472 281-5566 323-7022 326-7146 327-7162 329-7400 333-7445
BIT16   39-0    42-0    43-0    43-0    43-0
BIT17   39-0    42-0    43-0    43-0
BIT18   39-0    42-0    42-0    43-0
BIT19   39-0    42-0    42-0    43-0
BIT20   39-0    42-0    42-0    43-0
BIT21   39-0    42-0    42-0    43-0
BIT22   39-0    42-0    42-0    43-0
BIT23   39-0    42-0    42-0    43-0
BIT24   39-0    42-0    42-0    43-0
BIT25   39-0    42-0    42-0    43-0
BIT26   39-0    42-0    42-0    43-0
BIT27   39-0    42-0    42-0    43-0
BIT28   39-0    42-0    42-0    43-0
BIT29   39-0    42-0    42-0    43-0    44-0
BIT30   39-0    42-0    43-0    44-0    44-0
BIT31   39-0    43-0    44-0    44-0
BIUCSR  41-0    156-2030 156-2033 156-2037 156-2041
BIW.SR  301-6271 301-6277 302-6312 306-6435
BIW.EX  301-6276 303-6325
BIW.LP  301-6274 302-6302
BIW.LQ  301-6250 304-6371 304-6406
BIW.LS  303-6330 305-6417 305-6423 306-6447
BIW.NO  301-6254 305-6417
BIW.QK  301-6300 302-6302
BIW.SO  301-6255 304-6361 305-6422
BIW.TS  303-6326 305-6426 305-6427
BIWRCM  40-0     81-63   93-303  95-331  98-372  112-633 116-725 117-766 158-2051 301-6264 303-6331 304-6371 306-6433
BIWSTA  41-0     87-211  88-220  126-1122 126-1125 129-1147 130-1160 130-1163
BLAST   64-0     226-4065 229-4157 229-4163 231-4231 231-4240 236-4447 236-4451 237-4455 237-4457 237-4463 237-4470 246-4777 246-5002
```



```
        327-7162
BLOCK   19-0
BLRWR   64-0    234-4334 234-4350 236-4435 247-5006
BLSTB   64-0    236-4450 246-4777
BMAPDN  64-0    248-5025 248-5030
BNXCOP  64-0    323-7026 323-7027
BOARD2  73-0    80-47
BP      306-6450 315-6610 315-6614
BROOLP  290-5672 290-5702
BROOLX  290-5673 290-5704
BR01LP  290-5706 290-5717
BR01LX  290-5707 290-5721
BR02LP  290-5723 291-5734
BR02LX  290-5724 291-5736
BR03LP  291-5740 291-5751
BR03LX  291-5741 291-5753
BR04LP  291-5755 291-5766
BR04LX  291-5756 291-5770
BR05LP  291-5772 291-6003
BR05LX  291-5773 291-6005
BR06LP  292-6007 292-6020
BR06LX  292-6010 292-6022
BR07LP  292-6024 292-6035
BR07LX  292-6025 292-6037
BR08LP  290-5675 290-5677
BR09LP  290-5711 290-5713
BR10LP  290-5726 290-5730
BR11LP  291-5743 291-5745
BR12LP  291-5760 291-5762
BR13LP  291-5775 291-5777
BR14LP  292-6012 292-6014
BR15LP  292-6027 292-6031
BRARS   64-0    236-4436
BRCTS   64-0    234-4341 234-4352 236-4436 324-7121 327-7160
BREG    18-0    76-1    76-3    80-55   80-56   81-73   86-155  86-156  86-171  95-335  98-365  98-366  107-531 107-532
        108-552 108-553 113-642 114-666 115-712 115-713 116-742 116-743 117-754 117-755 119-1022 119-1023 122-1070 122-1071
        126-1120 129-1153 130-1162 133-1221 133-1223 133-1224 139-1323 148-1701 148-1702 156-2035 156-2036 160-2072 160-2075 289-5643
        299-6212 299-6213 299-6237 299-6240 301-6250 334-7771 335-7772
BRGADR  92-265  93-302  94-313  112-632 114-656 115-674 116-723 117-765 118-776 158-2050 299-6217
BRLV4   43-0
BRLV4R  43-0
BRLV4S  43-0
BRLV5   43-0    156-2032
BRLV5R  43-0    156-2040
BRLV5S  43-0
BRLV6   43-0
BRLV6R  43-0
BRLV6S  43-0
BRLV7   43-0
BRLV7R  43-0
BRLV7S  43-0
BROKE   42-0    87-207
BRTRY   64-0    236-4447 246-4775 247-5005
BTCNT   51-0    51-0    229-4147 230-4167 234-4335 236-4444
```

```
BTROTH   45-0     220-3731
BUF       18-0      18-0      84-133    86-161    91-264    92-268    92-271    93-306    93-310    93-312    94-321    94-322    98-355
          98-360   104-504   106-513   106-514   106-516   108-541   108-543   108-555   108-557   108-561   108-564   108-566   108-571   108-573
         108-601   109-604   110-610   110-612   110-614   110-616   110-620   110-622   110-624   114-647   115-706   115-710   116-730   116-732
         116-734   117-760   117-771   117-773   118-1003  118-1005  118-1007  118-1011  122-1053  122-1063  122-1065  124-1074  124-1100  126-1133
         131-1201  132-1204  132-1207  132-1211  132-1213  134-1230  136-1236  136-1246  136-1250  139-1317  140-1502  140-1504  140-1507  140-1511
         140-1513  140-1521  140-1532  140-1534  140-1536  142-1545  142-1550  143-1571  144-1575  145-1625  147-1650  148-1677  149-1707  149-1733
         151-1766  151-1776  153-2003  156-2026  158-2054  158-2056  158-2060  161-2100  161-2106  162-2121  166-2234  166-2240  168-2260  170-2301
         170-2302  172-2314  172-2321  173-2332  173-2335  174-2363  174-2364  175-2376  176-2423  177-2427  181-2501  183-2535  184-2543  184-2546
         186-2574  186-2600  188-2624  192-2763  192-2764  193-3023  194-3061  195-3077  195-3101  195-3107  196-3127  196-3144  196-3145  197-3174
         198-3211  198-3213  199-3222  199-3225  199-3227  201-3256  201-3260  201-3264  201-3270  202-3333  203-3360  203-3365  203-3366  209-3446
         209-3503  210-3515  210-3532  210-3540  211-3557  211-3561  211-3562  211-3577  212-3615  212-3620  213-3650  213-3651  214-3671  215-3674
         218-3713  218-3715  224-3774  224-3777  225-4041  225-4045  226-4060  226-4067  226-4073  227-4114  227-4121  227-4123  228-4141  228-4143
         228-4144  229-4150  230-4170  230-4176  230-4202  230-4213  231-4223  231-4224  231-4230  231-4231  231-4235  231-4236  231-4240  232-4260
         232-4264  232-4265  232-4270  233-4306  233-4316  233-4322  233-4325  234-4336  234-4343  234-4344  234-4347  234-4351  234-4353  234-4357
         234-4361  235-4411  235-4423  235-4424  236-4433  236-4451  237-4454  237-4457  237-4462  237-4466  237-4470  238-4516  238-4536  238-4541
         239-4552  239-4555  239-4561  240-4612  240-4621  240-4622  241-4636  241-4637  241-4644  243-4674  243-4675  243-4700  244-4722  245-4737
         246-4753  246-4766  246-5003  247-5007  248-5030  248-5041  249-5055  249-5067  249-5076  249-5100  250-5103  250-5106  250-5113  250-5120
         250-5122  250-5123  251-5131  251-5133  251-5137  251-5144  251-5155  251-5163  251-5164  252-5172  253-5215  253-5220  255-5226  257-5241
         259-5316  259-5377  259-5401  263-5423  264-5427  264-5431  264-5434  268-5463  268-5464  268-5467  273-5506  273-5510  273-5512  273-5514
         273-5515  273-5520  278-5537  279-5543  281-5560  282-5567  282-5572  282-5573  282-5574  282-5575  282-5576  282-5577  282-5600  282-5601
         282-5602  282-5603  282-5604  282-5606  282-5610  283-5613  283-5614  283-5616  283-5620  283-5621  283-5622  283-5623  283-5624  283-5625
         283-5626  283-5627  284-5630  284-5631  284-5632  289-5637  289-5641  290-5674  290-5675  290-5701  290-5702  290-5710  290-5711  290-5715
         290-5717  290-5725  290-5726  290-5732  291-5734  291-5742  291-5743  291-5747  291-5751  291-5757  291-5760  291-5764  291-5766  291-5774
         291-5775  291-6001  291-6003  292-6011  292-6012  292-6016  292-6020  292-6026  292-6027  292-6033  292-6035  293-6042  293-6047  293-6050
         293-6051  293-6053  293-6054  293-6060  293-6067  293-6070  293-6071  293-6072  293-6073  293-6074  294-6106  295-6132  296-6155  299-6238
         301-6244  301-6246  301-6271  301-6272  301-6274  301-6277  301-6300  302-6313  302-6315  302-6317  302-6321  302-6323  303-6336  303-6340
         303-6354  304-6376  304-6400  304-6414  306-6435  306-6436  306-6440  308-6457  308-6462  308-6464  308-6476  309-6500  309-6502  309-6510
         312-6532  312-6547  312-6551  312-6557  312-6561  313-6566  315-6614  315-6616  316-6642  317-6661  317-6663  317-6671  318-6735  318-6737
         319-6745  322-7013  322-7021  323-7027  323-7037  323-7052  323-7056  324-7060  324-7071  324-7075  324-7104  324-7106  324-7110  324-7121
         324-7123  324-7125  326-7143  326-7146  331-7440  333-7725  335-7775  335-7775
BUF.56   65-0      65-0
BUF.BC   65-0      65-0       65-0     212-3611  243-4701  244-4713  249-5052  249-5070  250-5102  250-5111  251-5154  253-5176
BUF.BP   64-0      64-0      212-3616  212-3622  230-4201  236-4431  244-4716  249-5073  273-5507  307-6451  318-6676  322-7010  323-7033  324-7063
BUF.DL   65-0      65-0       65-0     140-1525  307-6454
BUF.EC   65-0      65-0
BUF.ED   65-0      65-0
BUF.GP   65-0      65-0      212-3614  230-4212  235-4410  243-4706  246-4771  249-5065  249-5077  251-5127  253-5201
BUF.HH   64-0      64-0      233-4311  233-4311  240-4600  240-4600  250-5121  322-6776  323-7054  323-7054  326-7150
BUF.HL   64-0      64-0      233-4311  240-4600  250-5117  322-6775  323-7054  326-7150  326-7150
BUF.LL   65-0      65-0       65-0      65-0     225-4042  226-4061
BUF.NL   64-0      64-0      224-3776  226-4052  226-4066  229-4154  229-4161  231-4226  234-4331  236-4446  237-4452  237-4465  246-4776  273-5506
         327-7156
BUF.SD   64-0      64-0      196-3134  225-4043  232-4250  235-4417  236-4427  239-4547  239-4571  243-4677  244-4727  253-5175  318-6731
BUF.ST   64-0      64-0      211-3565  230-4203  234-4342  234-4360  235-4367  236-4434  236-4445  240-4624  246-4747  247-5004  248-5024  248-5027
         251-5140  323-7023  324-7120  327-7157
BUF.TA   64-0      64-0      251-5151  322-7002  322-7005  326-7152
BUF.U1   65-0      65-0      212-3612  237-4505  249-5063
BUF.U2   65-0      65-0       65-0     212-3613  237-4506  248-5021  248-5023  249-5064  249-5075
BUF.UA   64-0      65-0      202-3342  248-5033  249-5071  251-5143
BUF.US   65-0      65-0       65-0     202-3343  248-5034  249-5072  251-5136
BUFBEG   65-0      65-0      276-5532  276-5533
BUFEND   65-0      65-0
BUFLMT   45-0     225-4031  225-4032
```



```
BUFP00   49-0      49-0
BUFP01   49-0      49-0
BUFP02   49-0      49-0
BUFP03   49-0      49-0
BUFP04   49-0      49-0
BUFP05   49-0      49-0
BUFP06   49-0      49-0
BUFP07   49-0      49-0
BUFP08   49-0      49-0
BUFP09   49-0      49-0
BUFP10   49-0      49-0
BUFP11   49-0      49-0
BUFP12   49-0      49-0
BUFP13   49-0      49-0
BUFP14   49-0      49-0
BUFP15   50-0      50-0
BUFP16   50-0      50-0
BUFP17   50-0      50-0
BUFP18   50-0      50-0
BUFP19   50-0      50-0
BUFP20   50-0      50-0
BUFP21   50-0      50-0
BUFP22   50-0      50-0
BUFP23   50-0      50-0
BUFP24   50-0      50-0
BUFP25   50-0      50-0
BUFP26   50-0      50-0
BUFP27   50-0      50-0
BUFP28   51-0      51-0
BUFP29   51-0      51-0
BUFP30   51-0      51-0
BUFP31   51-0      51-0
BUFP32   51-0      51-0
BUFP33   51-0      51-0
BUFP34   51-0      51-0
BUFP35   51-0      51-0
BUFP36   51-0      51-0
BUFP37   51-0      51-0
BUFP38   51-0      51-0
BUFP39   51-0      51-0
BUFP40   51-0      51-0
BUFP41   51-0      51-0
BUFP42   51-0      51-0     140-1515 240-4627
BUFP43   51-0      51-0
BUFP44   51-0      51-0
BUFP45   51-0      51-0      51-0      51-0
BUFP46   51-0      51-0
BUFP47   51-0      51-0      51-0      51-0
BUFP48   51-0      51-0
BUFP49   51-0      51-0      87-201   268-5466 334-7750
BUFP50   51-0      51-0      51-0
BUFP51   51-0      51-0
BUFP52   51-0      52-0     230-4173
BUFPTR   49-0      49-0
```

LSCS FORM=QUAD

CROSS REFERENCE TABLE (CREF V01-08)

```
BUFR1.   65-0     65-0     65-0    275-5530 275-5531
BUFR2.   65-0    125-1106 125-1107
BW.S01 304-6362 304-6367
C$TYPE     1-0      1-0      8-0      8-0      8-0     12-0     12-0     12-0     13-0     13-0     13-0     14-0     14-0     15-0
          15-0     15-0     15-0     15-0     15-0     18-0     18-0     18-0     20-0     21-0     22-0     22-0     22-0     25-0
          27-0     29-0     30-0     30-0     38-0     45-0     45-0     45-0     46-0     46-0     46-0     51-0     51-0     51-0
          51-0     59-0     63-0     64-0     65-0     65-0     65-0     65-0     67-0     67-0     70-0     71-0     71-0     71-0
          73-0     74-0     74-0     74-0     74-0     75-0     76-0     76-0     76-0     76-3     76-3     76-3     76-3     77-11
          77-11    77-11    77-15    77-15    77-15    77-16    79-26    79-27    79-31    80-37    80-45    80-54    80-55    80-55
          80-55    82-75    82-75    82-75    82-75    83-105   83-111   83-112   84-114   84-116   84-122   84-125   84-130   85-142
          85-147   85-153   85-153   85-153   91-252   91-253   102-456  102-456  102-456  102-456  102-456  102-460  102-466  103-475
         104-500  106-530  122-1072 124-1075 124-1077 124-1104 126-1110 126-1115 126-1115 126-1115 128-1137 128-1137 128-1137 128-1145
         128-1146 129-1147 129-1147 129-1147 132-1210 133-1215 133-1216 133-1220 133-1222 133-1222 133-1222 134-1233 134-1234 138-1300
         138-1300 139-1315 139-1321 139-1322 139-1325 140-1514 140-1515 142-1541 142-1541 143-1572 145-1604 145-1624 148-1701 148-1704
         148-1706 152-2002 153-2004 155-2014 155-2023 155-2023 160-2072 160-2072 160-2072 176-2424 191-2725 191-2730 193-3007 196-3136
         201-3305 202-3344 203-3355 206-3410 207-3414 212-3612 212-3614 220-3725 220-3725 220-3725 221-3741 221-3743 222-3763 226-4070
         230-4206 230-4211 230-4212 232-4241 232-4247 232-4254 235-4404 235-4407 235-4410 237-4503 237-4510 238-4512 240-4627 243-4704
         243-4706 246-4771 248-5014 248-5016 248-5022 248-5034 249-5057 249-5065 249-5075 249-5075 249-5075 251-5127 251-5136 253-5177
         253-5201 256-5232 259-5333 259-5342 259-5352 268-5466 268-5466 278-5536 278-5541 278-5542 278-5542 278-5542 282-5567 285-5633
         286-5633 287-5634 287-5634 288-5634 288-5634 288-5634 289-5634 296-6170 297-6201 297-6201 300-6242 300-6242 300-6242 301-6242
         306-6450 329-7400 329-7404 329-7404 330-7426 330-7426 331-7440 332-7445 333-7445 333-7734 334-7750 334-7750 334-7750 334-7760
CCLASS   45-0    334-7757
CCSRWO   48-0     48-0
CCSRW1   48-0     48-0     48-0     48-0    149-1712 149-1722
CDONE    12-0     95-337   96-340   108-552  115-701  116-736  159-2062 290-5675 290-5702 290-5711 290-5717 290-5726 291-5734 291-5743
         291-5751 291-5760 291-5766 291-5775 291-6003 292-6012 292-6020 292-6027 292-6035 293-6060 298-6201 298-6202 298-6203 298-6204
         298-6205 298-6206 301-6300
CF.ATN   50-0     50-0    200-3244
CF.MSC   49-0     50-0
CF.THS   49-0     50-0    232-4275 233-4277
CFLAGS   50-0    214-3671
CHGFLG   69-0
CHGMOD   69-0
CHKBDR 289-5651 297-6170 297-6170 301-6256
CHKPAR   98-353  112-630  114-654  115-672  116-720
CKWREA 227-4122 227-4124
CKWRER 194-3037 227-4123 322-7007
CKWROM 227-4121 229-4155 253-5212
CLIMIT   49-0     49-0    140-1510 147-1664
CLRERR   90-251
CLRTST   79-31    80-34    82-75    82-75    82-75
CMDCOF   47-0     47-0    142-1554 142-1556 148-1705 150-1743 196-3134
CMDLEN   47-0     47-0     48-0     48-0    131-1171 148-1703
CMDLIM   45-0     52-0    140-1511
CMDO     48-0    149-1723
CMDPOF   47-0     48-0    150-1743
CMDPTR   47-0     47-0
CMODEL   45-0     45-0     45-0    268-5471 271-5477
CMPBUF   51-0     51-0    140-1526 243-4703 244-4717
CN.ERR   19-0    142-1541
CNT.LN   52-0    201-3313
CNTFLG   49-0     50-0     50-0    200-3236 214-3672 233-4300
CNVTV1   48-0     48-0    179-2442 179-2443 192-2762 192-2762 192-2762 210-3537 210-3537 210-3537 266-5447
CNVTV2   48-0     48-0    179-2431 179-2432
```


CROSS REFERENCE TABLE (CREF V01-08)

```
CNVTV3   48-0     48-0    179-2430 179-2452 289-5640 289-5641 299-6231 301-6245 301-6246
CNVTV4   48-0     48-0    179-2456 192-2762 210-3537 289-5641 299-6232 301-6246
CODVER   46-0     46-0     46-0     87-202  138-1274 138-1275 202-3335 214-3666 268-5470 268-5471
COMPLT   69-0
CON.ER 334-7766 335-7774 335-8000
CON.ST   50-0    126-1132 132-1210 134-1231 140-1530
CONT     24-0
CONTCD   69-0
COPY4     1-0
CPE      12-0     12-0     12-0     83-106   83-110   83-112   84-121   84-123
CR       18-0     18-0     18-0
CRDY     24-0
CRI      17-0     76-4     80-37    80-45    80-50    80-57    80-60    81-72    83-105   83-113   84-115   84-124   84-126   84-130
         85-143   89-240   89-241   95-327   96-343  100-444  103-473  104-500  107-534  107-535  108-554  108-574  114-645  114-653
        114-656  114-665  115-674  115-711  116-722  116-741  334-7760
CRY      13-0     99-401  147-1667 162-2135 162-2141 179-2445 179-2464 191-2724 192-2736 192-2753 192-2755 197-3160 197-3163 210-3526
        219-3723 225-4014 249-5054 250-5106 250-5121 251-5163 257-5263 259-5332 259-5371 262-5415 264-5434 269-5472 270-5475 280-5555
        295-6136 303-6343 304-6365 304-6403 306-6442 322-6772
CSERR    69-0
CSR      14-0
CTMOUT   46-0    214-3665
CYCLE    36-0
CYLSTR   45-0    192-2763
D.E.C.    1-0
D.STPA 333-7445 333-7725
DATEND   65-0     65-0     66-0
DATT     59-0    142-1546 163-2157 196-3131
DBAR     17-0
DCERR    69-0
DCLASS   45-0    216-3701
DCLK     14-0
DCMASK   62-0
DCN.ST   69-0
DCN.TT   69-0
DCRD     18-0     18-0     18-0
DCRS     18-0     18-0     18-0
DDC      22-0
DDD      22-0
DER      17-0     80-54    85-144   88-215   89-234
DERR     59-0    196-3126 211-3550 211-3562 211-3601 213-3651 232-4267 245-4734 245-4741 247-5013
DEVCL    45-0
DFAIL    21-0     21-0     80-45    80-50    83-105   83-113   84-126  102-464  334-7761
DI       73-0     87-213
DIAG     73-0
DIINTR 126-1120 127-1134
DISCON   69-0
DISM     48-0    312-6532 312-6532 312-6550 313-6572
DISN     48-0    312-6532 312-6542 312-6546 313-6571
DIV512 224-4005 227-4127 257-5255
DIVA   280-5551 280-5556
DIVB   280-5547 280-5552 280-5554
DIVC   280-5553 280-5555
DIVD   179-2441 280-5547
DIVDO  179-2450 179-2455 280-5546
```

```
DIVX    280-5550 280-5555
DIVXIT  280-5550 280-5557
DM.BEG   65-0     65-0     65-0     262-5412 262-5413 267-5456 268-5462 277-5534 277-5535 333-7745 333-7746
DM.RD   289-5653 296-6142
DM.RLP  296-6144 296-6166
DM.WR   301-6260 306-6430 306-6445
DMATST  136-1234 136-1246
DMBEG    19-0    220-3730 221-3762 222-3763 333-7732 333-7736
DMBPC    45-0
DMODE    19-0    142-1542 145-1606 156-2045 164-2175 177-2424 196-3137 207-3415 221-3737 230-4172 267-5461 289-5652 301-6257
DMODT    45-0    204-3373
DMOVH    45-0
DMOVL    45-0
DMREG0   47-0     47-0     262-5407 262-5421 263-5425 264-5436 266-5446 267-5454 333-7741
DMREG1   47-0     47-0     262-5410 264-5430 266-5450
DMREG2   47-0     47-0     140-1512 262-5416 264-5435 307-6452
DMREG3   47-0     47-0     262-5411
DMREG4   47-0     47-0
DMREG5   47-0     47-0
DMREG6   47-0     47-0
DMREG7   47-0     47-0
DMSTR    45-0    262-5412
DMTEMP   49-0     49-0
DOT     306-6450 312-6540 316-6634 319-6751
DOT1    306-6450 319-6752 319-6757
DOT2    306-6450 319-6751 319-6753
DPF      14-0
DPROC    12-0
DRAVL    59-0    124-1074 142-1552 163-2157 174-2357 175-2371 176-2404 181-2500 185-2554 186-2576 209-3444 213-3650
DRDUP    59-0    124-1074 163-2157 176-2404 176-2414 181-2500 185-2563 186-2576 209-3444 213-3650
DRDY     25-0
DRINIT   21-0     21-0     126-1114 145-1617 145-1620 145-1621 146-1634 148-1676
DRV.AT   63-0     63-0     201-3255 201-3256
DRV.AV   62-0     63-0     188-2642 193-2773 209-3471
DRV.C1   62-0
DRV.C2   62-0
DRV.C3   62-0
DRV.C4   62-0
DRV.DB   62-0
DRV.DD   62-0    176-2407 209-3474
DRV.DE   62-0
DRV.DF   62-0     62-0
DRV.DR   61-0
DRV.EL   61-0
DRV.FO   62-0
DRV.OA   61-0
DRV.PE   62-0     62-0
DRV.PS   61-0
DRV.RE   62-0     62-0
DRV.RR   61-0
DRV.RU   61-0    175-2367 176-2406 209-3474
DRV.S1   62-0     62-0
DRV.S2   62-0     62-0
DRV.S3   62-0     62-0
```



```
DRV.S4   62-0     62-0
DRV.S7   62-0
DRV.SN   62-0
DRV.SR   61-0
DRV.SU   61-0     63-0     202-3323
DRV.U1   61-0     61-0     186-2572
DRV.U2   61-0     61-0     227-4107
DRV.U3   61-0     61-0     227-4107
DRV.U4   61-0     61-0     227-4107
DRV.UM   63-0    186-2601 187-2612 187-2613 188-2626 200-3250 201-3260
DRV.W1   62-0
DRV.W2   62-0
DRV.W3   62-0
DRV.W4   62-0
DRV.WE   62-0     62-0
DRVCLR   69-0
DRVOL    59-0    124-1074 142-1552 163-2157 181-2500 185-2554 186-2576 209-3444 213-3650
DSER     14-0
DSK.LN   52-0    203-3357
DSLEDS   84-130   89-236
DSR      14-0
DSTSH    45-0    220-3735 268-5465
DSTSL    45-0    220-3734 221-3755 268-5462 268-5465 333-7746
DTEMP1   50-0     50-0
DTMP     17-0
DUP.LN   52-0    200-3246
DUPVC    45-0    165-2224
DXFC     19-0    143-1564 207-3415 207-3416 221-3737 221-3740 262-5422 266-5453 267-5461 333-7736
E.RPE2  334-7764 334-7771
E.WPE2  334-7760 335-7772
E311    306-6450 312-6535 313-6600
E335    306-6450 312-6555 312-6561
E345    306-6450 313-6566 313-6575
E350    306-6450 312-6541 313-6577
E900    306-6450 309-6517 317-6665 318-6675
E910    306-6450 318-6702 318-6727
E915    306-6450 318-6715 318-6716
E917    306-6450 318-6714 318-6716
E920    306-6450 318-6715 318-6717
ECC      18-0
ECCOA   306-6450 308-6471 308-6476
ECC1    306-6450 309-6477
ECC1A   306-6450 309-6506 309-6512
ECC1B   306-6450 309-6505 309-6512
ECC2    306-6450 309-6500 309-6502 309-6510 309-6511 311-6520
ECC2A   306-6450 311-6523 311-6527
ECC9.1   48-0     48-0     48-0
ECC9.2   48-0     48-0     48-0
ECC9.3   48-0     48-0
ECCC    211-3571 234-4355 263-5423 306-6450 307-6450 323-7031
ECCCA   306-6450 308-6457 308-6465
ECCCB   306-6450 308-6457 308-6467
ECCMSK   73-0
ECHOCD   69-0
```

```
ECMOV9  306-6450 313-6570 313-6576 317-6670
ECMOVE  266-5447 306-6450 317-6666 317-6670 317-6672
ECMOVX  306-6450 317-6671 317-6674
ECOUNT   49-0     49-0    235-4375 263-5424 309-6514 312-6534 315-6622 316-6644 318-6701
ECSUMH   73-0
ECSUML   73-0
EDSEED   45-0    122-1054 122-1060 136-1237 136-1243 137-1257 230-4210 235-4406 243-4671 289-5635 301-6242 323-7034
EF.BBR   59-0    239-4560 240-4611 240-4620 240-4622
EF.BBU   59-0    239-4560 240-4612 240-4621
EF.LOG   59-0    202-3333 239-4560
ELEV     61-0    161-2100 162-2132
ELOC     48-0    309-6513 315-6624 317-6650 318-6677
ELPM     48-0     48-0    311-6523 312-6554 313-6570
ELPN     48-0    308-6455 308-6471 308-6472 311-6521 311-6523 311-6524 312-6540 313-6567 313-6575 315-6610 317-6665
ELPO     48-0     48-0     48-0    311-6524 312-6541 312-6560 313-6576
ELPP     48-0     48-0     48-0     48-0     48-0    315-6607 315-6625 316-6632 317-6666
ENCACH   65-0     65-0
ENDCD    69-0     69-0
EQ       13-0     47-0     48-0     49-0     59-0     65-0     67-0     77-5    101-450  101-454  137-1274 139-1500 142-1545 146-1630
        147-1662 148-1704 149-1711 149-1723 150-1743 150-1743 154-2005 162-2114 162-2132 163-2164 163-2166 164-2171 164-2171 164-2173
        164-2173 175-2367 175-2371 176-2377 176-2404 176-2406 180-2473 182-2502 184-2544 188-2623 189-2645 189-2646 189-2647 189-2650
        189-2652 189-2657 191-2720 191-2730 192-2757 192-2760 192-2767 192-2767 192-2767 192-2767 193-3000 193-3002 193-3005 193-3010
        194-3034 194-3035 194-3035 194-3042 194-3042 194-3047 196-3122 196-3122 196-3144 196-3144 196-3144 197-3156 197-3165 197-3173
        197-3175 198-3210 202-3326 202-3326 202-3335 205-3401 205-3401 205-3401 205-3401 209-3472 209-3474 211-3542 211-3575 212-3630
        212-3632 212-3633 212-3634 212-3635 214-3666 216-3676 216-3677 217-3703 217-3705 217-3706 217-3707 217-3710 223-3774 224-3776
        224-4005 225-4011 225-4023 226-4054 226-4072 226-4074 226-4105 227-4126 230-4204 231-4224 232-4272 233-4311 233-4312 233-4316
        236-4447 240-4600 240-4602 240-4623 246-4745 246-4777 247-5005 248-5017 248-5023 251-5153 251-5157 257-5232 257-5255 258-5273
        268-5465 269-5472 273-5506 289-5641 289-5657 301-6246 312-6532 312-6532 317-6665 322-6765 322-6773 322-7001 323-7054 324-7062
        324-7072 324-7072 324-7073 326-7147 326-7150 327-7162 333-7725 333-7747 333-7747 333-7747
ER.BCA   67-0
ER.BP1   67-0    114-650
ER.CID  201-3313 202-3327 202-3334
ER.DMX   67-0    262-5415 267-5455
ER.HTO   67-0    145-1616
ER.INT   67-0    156-2026
ER.IWR   67-0    155-2021
ER.LOG  200-3244 201-3262
ER.MER   67-0     80-40
ER.MRR   67-0    196-3142 226-4102 259-5356
ER.MST   67-0    289-5637 301-6244
ER.NIM   67-0    165-2222
ER.PRD   67-0
ER.PWR   67-0    147-1653
ER.RAP   67-0    101-446  101-452  104-476
ER.ROP   67-0    103-472
ER.RP2   67-0    114-662
ER.RRD   67-0    149-1714 150-1752
ER.RTO   67-0     67-0     67-0
ER.RWR   67-0    151-1770
ER.SAP   67-0    101-445  101-451  335-7776
ER.SOP   67-0    335-7777
ER.SRN  202-3326 202-3345
ER.SRP   67-0     67-0    334-7762 334-7763 334-7764 335-7773
ER.STP   67-0    299-6236
```



```
ER.TMO   67-0
ER.UID  201-3303 201-3314 203-3357
ER.VCI   67-0    164-2176
ER.WP2   67-0    113-637
ERECOV   69-0
ERET    306-6450 318-6730 318-6734
ERR      35-0     38-0
ERR00    38-0     77-10    77-12
ERR01    38-0     79-27
ERR02    38-0     80-46
ERR03    38-0
ERR04    38-0     80-34
ERR05    38-0     79-31
ERR06    38-0     80-36
ERR07    38-0     38-0     80-43
ERRB1    79-30    79-33    80-35    80-36    80-44    80-45    88-217
ERRB1A   79-26    99-373   99-374   99-375   99-376   99-377   99-400   99-401   99-402   99-403   99-405   99-406   99-407   99-410
         99-414   99-415   99-417   99-420  100-442  100-444
ERRB1B   80-41    80-42    80-43   108-552  108-554  110-625  113-641  113-642  114-652  114-653  114-664  114-665  115-711  115-716
        116-741  116-746  118-1012 119-1021 119-1024 120-1034 122-1055 122-1061 122-1066
ERRB1E   79-27    79-27    83-106   83-110   83-111   83-112   84-121   84-122   84-123  101-447  101-450  101-453  101-454  102-463
        103-470  103-473
ERRB1L   80-34   129-1152
ERRB1M   80-37    95-323   96-346  159-2061 159-2064
ERRB1O   79-31   128-1142 128-1145 133-1222 134-1225 136-1240 136-1244 136-1251 137-1260
ERRB1Q   80-36
ERRB1T   80-42    86-165   95-330   96-345  115-703  116-740
ERRB1Z   79-27   173-2330 238-4517 238-4524
ERRB2E   80-46   104-477  106-514  106-517
ERRBIT   73-0     80-51    81-61   160-2073
ERREG    37-0
ERRINI   60-0
ERRINP   60-0    246-4745 246-4746
ERRIP    59-0    124-1074 173-2336 211-3602 247-5011
ERRRR   306-6450 315-6602 315-6603 315-6606 316-6632 316-6635 316-6640 316-6642 318-6705 318-6734
ERRRTC   45-0    246-4757
ERRSET   80-45    80-50    80-51
ERRVEC   60-0
ETST1    12-0
EVAL     48-0    309-6516 316-6645 318-6700
F       306-6450 317-6653 317-6663
FAILUR   48-0     48-0     48-0     48-0    104-502  106-511  140-1531 142-1540 155-2022 160-2071 201-3304 289-5636 299-6235 301-6243
FAIRCT   61-0    195-3077
FCT.MD   45-0    212-3624
FCT.V1   45-0     45-0    212-3627
FCT.V2   45-0    212-3630
FDCLK    22-0
FDIAG    61-0    189-2651 202-3316
FLAG     35-0    151-1766 151-1771
FLOAT    18-0
FM.BAD   58-0    201-3272 232-4246 232-4251
FM.CNT   58-0    140-1534 201-3271
FM.DSK   58-0    201-3273 236-4441 239-4565 239-4575
FM.SDI   58-0
```

CROSS REFERENCE TABLE (CREF V01-08)

```
L.SDI5   58-0
L.SEQ    57-0
L.UHVR   57-0
L.UNIT   57-0     202-3324
L.UNTO   57-0      57-0     201-3314
L.UNT1   57-0      57-0
L.UNT2   57-0      57-0
L.UNT3   57-0
L.USVR   57-0      57-0
L.VSEO   57-0      57-0
L.VSE1   57-0
LAB       1-0
LADD     15-0      92-267    92-272    98-370   108-555   110-607   289-5654 293-6076 295-6117 296-6144 301-6261 303-6333 303-6350 304-6373
        304-6410 306-6430
LATE     14-0
LBNMSK   45-0     180-2473  189-2645  192-2760  225-4011  225-4023  251-5157  322-6773
LBNSTR   45-0     194-3035
LBWR     15-0      81-66     93-312   113-636   116-734   117-773   158-2060  301-6274  303-6340  304-6400  306-6440
LCOM     15-0      81-64     93-304    94-315   108-564   110-612   113-634   114-660   115-676   116-726   117-767   118-1000 158-2052 289-5657
        293-6102 295-6124  296-6146  299-6221  301-6266  303-6332  303-6347  304-6363  304-6372  305-6424  306-6434
LDR2R1  191-2720 192-2757  193-3000  193-3005  194-3034  197-3156  197-3165  197-3173  198-3211  233-4311  240-4600  240-4602  242-4656  322-6765
        322-7001 323-7054  324-7062  326-7147
LED1     22-0      22-0      80-50     89-236   126-1110  126-1114  129-1154  145-1607  299-6241
LED2     22-0      22-0      80-45    126-1114  129-1154
LED4     22-0      22-0      80-45     80-50    126-1110  126-1112  129-1154
LED8     22-0      22-0     126-1110  126-1112  126-1114
LEDS     22-0      84-130   100-444   103-473   138-1301
LESS     12-0     162-2137
LF.CON   58-0     201-3263  239-4566
LF.SNR   58-0     201-3264
LF.SUC   58-0     201-3263  203-3350  236-4442
LFAIL    50-0     139-1317  140-1527
LG.BAD  201-3273 202-3341
LG.CNT  201-3272 201-3304
LG.DSK  201-3274 202-3346
LG.SDI  201-3275
LGADR    65-0     137-1264  238-4515  238-4522  282-5570  315-6612  316-6636  316-6640  317-6651  318-6736  319-6746  319-6754
LGCKSV   51-0      51-0     137-1273
LM       48-0     312-6533  313-6562  313-6573
LN       51-0      51-0     311-6530  312-6535  313-6564  315-6601  315-6604  315-6626  317-6647  317-6664
LNERR    70-0
LNPAG    15-0     108-571   110-614   114-644   114-646   297-6175  297-6176
LO.OFF  101-453  101-453   101-454
LOBYT    45-0      45-0      45-0      46-0      60-0      87-203   125-1106  138-1274  142-1543  142-1560  144-1572  149-1730  154-2005  155-2024
        156-2044 179-2434  179-2440  180-2473  184-2540  189-2645  192-2760  203-3362  203-3367  212-3610  212-3615  219-3723  225-4011  225-4023
        230-4214 235-4412  243-4707  246-4762  246-4772  249-5100  251-5130  251-5157  253-5202  261-5402  262-5412  268-5470  274-5522  274-5524
        274-5526 275-5530  276-5532  277-5534  281-5561  282-5605  282-5607  283-5617  322-6773  323-7056  324-7106  324-7123  333-7745  333-7746
LOG.LN   52-0      52-0     200-3234
LOGCOD   45-0
LOGEND   52-0      52-0
LOGLEN   49-0      49-0     147-1646  202-3337
LOGPKT   52-0      52-0     147-1644  154-2010  164-2174  165-2221  200-3233
LONIB    45-0     147-1672  149-1733  164-2172  164-2173  176-2422  189-2656  211-3575  212-3606  322-6763
LOPAR   101-453  329-7403
```


CROSS REFERENCE TABLE (CREF V01-08)

```
LRGADR   81-62     92-272
LSB      12-0      98-361    99-375    99-406   112-631   114-655   116-721   126-1132  134-1232  138-1311  175-2371  192-2745  193-3001  193-3006
        230-4170 230-4204  231-4224  233-4316  259-5324  281-5565  293-6052
LT400    73-0
LV1SV1   50-0      50-0
LV1SV2   50-0      50-0
LV2CNT   60-0
LW       44-0      86-161    91-264    92-273   303-6334  303-6351  304-6374  304-6411
M        48-0     311-6531  312-6532  312-6543  313-6574
M.ABRT  170-2274 170-2276
M.ACCH  190-2670 190-2671
M.ACCL  190-2670 190-2670
M.AVAH  174-2344 174-2345
M.AVAL  174-2344 174-2344
M.CCDH  170-2270 170-2271
M.CCDL  170-2270 170-2270
M.CMPH  190-2662 190-2663
M.CMPL  190-2662 190-2662
M.DAP   174-2346 174-2346
M.ERSH  191-2672 191-2673
M.ERSL  191-2672 191-2672
M.FLUH  170-2272 170-2273
M.FLUL  170-2272 170-2272
M.GCST  183-2515 183-2517
M.GUSH  188-2617 188-2620
M.GUSL  188-2617 188-2617
M.ONLH  208-3424 208-3424
M.ONLL  208-3424 208-3425
M.RDH   190-2664 190-2665
M.RDL   190-2664 190-2664
M.RPLH  191-2674 191-2675
M.RPLL  191-2674 191-2674
M.SCCL  214-3653 214-3653
M.SUCH  208-3422 208-3422
M.SUCL  208-3422 208-3423
M.WRH   190-2666 190-2667
M.WRL   190-2666 190-2666
MAP.CH   63-0      63-0     144-1574  259-5354  259-5374
MAP.MO   63-0      63-0     249-5043  257-5234  258-5314  259-5341
MAP.M1   63-0      64-0     249-5045  257-5235  259-5315  259-5343
MAP.ND   64-0      64-0
MAP.NX   63-0      63-0     249-5047  258-5273  259-5375
MAP.OF   64-0      64-0     248-5031  257-5240  257-5257  257-5267
MAP.RD   64-0      64-0     248-5035  257-5236  259-5364
MAP.S1   64-0      64-0     249-5051  257-5270
MAP.ST   64-0      64-0     196-3136  197-3153  226-4077  257-5233
MAP.UR   64-0      64-0     248-5037  257-5256  257-5265  259-5345  259-5361
MAP.V0   64-0      64-0     249-5042  257-5242  259-5320  259-5334  259-5365  259-5372
MAP.V1   64-0      64-0     249-5044  257-5244  259-5321  259-5337  259-5366  259-5373
MAP.V2   64-0      64-0     257-5246  259-5327
MAP.V3   64-0      64-0     257-5250  259-5330
MAPENB   45-0
MAPFLG   51-0      51-0     248-5040  259-5376
MAPMSC   45-0
```

```
MAPMSH   45-0
MAPMSK   45-0      259-5326
MAPSAV   51-0       51-0     244-4710 244-4714 259-5351 259-5360
MAPVAL   45-0      226-4072 248-5023 258-5273
MAXSTC  219-3722 333-7737
MCP.IO   52-0      147-1643
MCP.LN   52-0       52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0      52-0
         52-0       52-0      52-0      52-0      52-0      52-0      52-0     146-1635 154-2011 184-2542 198-3206
MCP.RD   52-0      149-1713
MD.ALL   58-0      174-2344
MD.CMP   58-0      190-2664 190-2666 196-3121
MD.CSE   58-0      170-2270 170-2272 174-2344 188-2617 190-2662 190-2664 190-2666 190-2670 191-2672 191-2674 208-3422 208-3424
MD.ERR   58-0      190-2666 191-2672 230-4220
MD.EXC   59-0      174-2344 208-3422 208-3424
MD.EXP   58-0      162-2114 170-2270 170-2272 190-2662 190-2664 190-2666 190-2670 191-2672 191-2674
MD.FEU   58-0
MD.FKC   59-0      214-3653
MD.IMF   59-0      208-3424 211-3566
MD.NOV   58-0      238-4541
MD.NXU   58-0      188-2617 188-2623
MD.PRI   59-0      191-2674
MD.RIP   58-0      208-3424 209-3476
MD.SAV   58-0
MD.SCH   58-0      170-2270 190-2662 190-2664 190-2670
MD.SCL   58-0      170-2270 190-2662 190-2664 190-2670
MD.SEC   58-0      170-2270 170-2272 190-2662 190-2664 190-2666 190-2670 234-4354
MD.SEQ   58-0      190-2666 191-2672
MD.SER   58-0      170-2270 170-2272 190-2662 190-2664 190-2666 190-2670 191-2672 234-4362
MD.SHD   58-0      209-3466
MD.SPD   58-0      174-2344 175-2371 211-3561
MD.SSH   58-0      170-2270 170-2272 190-2662 190-2664 190-2666 190-2670 191-2672
MD.SWP   58-0      208-3422 208-3424 210-3507
MD.VOL   58-0
MD.WBN   58-0      190-2666 191-2672
MD.WBV   58-0      190-2666 191-2672
MD512    45-0      212-3610 212-3623
MEMSZ    45-0       65-0     144-1576 259-5346 259-5347
MERR     12-0       95-323    96-346  158-2061 158-2064 290-5672 290-5677 290-5706 290-5713 290-5723 290-5730 291-5740 291-5745 291-5755
        291-5762 291-5772 291-5777 292-6007 292-6014 292-6024 292-6031 293-6062 294-6105 295-6127 296-6154 303-6326 303-6344 303-6357
        304-6366 304-6402 306-6444
MINUTE   45-0      140-1504
MLN20   168-2254 170-2274 183-2515
MLN40   168-2255 170-2306 191-2711 214-3655
MLN44   168-2256 208-3432
MLNTST  168-2254 168-2255 168-2256 168-2257 168-2257
MODNUM   46-0       46-0     138-1274 138-1275
MP       73-0       87-214
MRQUE    49-0       49-0     264-5440 333-7735 333-7742
MSB      13-0       91-260    98-420  102-464  147-1662 149-1723 162-2114 184-2544 211-3575 230-4174 234-4333 236-4430 246-4745 246-4777
        247-5005 303-6331 305-6422 324-7073
MSCPLN   51-0       51-0     149-1716 168-2257 209-3443 209-3457
MUL2RA  319-6743
MULBR2  306-6450 312-6555 318-6735 319-6756
MULBYT  225-4020 225-4023 281-5560
```



```
MULR2   306-6450 315-6612 318-6737 319-6746
MULR2X  306-6450 316-6643 317-6654 319-6741
MULT    179-2434 179-2440 281-5562
MULTA   281-5563 281-5566
MULTB   281-5562 281-5564
MWQUE    49-0       49-0     333-7734 333-7743
NACLO    12-0       12-0      13-0
NBCB     65-0       65-0      65-0     225-4036
NBIBAD   12-0
NBUFR    65-0       65-0      65-0     140-1520 230-4173 240-4627
NCDONE   13-0       95-324    95-325   108-554  159-2063 290-5676 290-5703 290-5712 290-5720 290-5727 291-5735 291-5744 291-5752 291-5761
        291-5767 291-5776 291-6004 292-6013 292-6021 292-6030 292-6036 293-6061 302-6301
NCPE     13-0       13-0      13-0     83-111   84-122
NCRY     12-0       99-403   128-1141 128-1142 162-2126 179-2454 192-2751 193-3017 197-3170 211-3553 225-4032 231-4230 231-4235 235-4377
        251-5145 259-5347 268-5464 278-5540 280-5552 289-5663 289-5670 293-6063 296-6151 297-6174 301-6267 302-6310 305-6426
NCSR     14-0
NDCLK    14-0
NDPF     14-0
NDSER    14-0
NDSR     14-0
NEG      13-0       85-143    89-237   99-374   99-401  100-444  104-502  106-513  106-524  137-1262 149-1720 168-2261 286-5633 289-5651
        295-6120 301-6256 304-6367 305-6417 306-6450 306-6450 308-6467 309-6504 309-6506 312-6553 313-6566 317-6657 317-6671 318-6715
        319-6741 319-6742 319-6745 319-6752 324-7076
NENDCD   69-0
NEQ      12-0      308-6476
NETST1   13-0
NFTEST   13-0       13-0      85-142
NLATE    14-0
NLSB     13-0       99-373    99-402   99-414   99-415  115-673  163-2164 175-2367 176-2406 188-2623 209-3474 228-4140 230-4176 234-4336
        245-4737 273-5517 293-6041 303-6346
NMERR    13-0      115-711   116-741  156-2035 290-5704 290-5721 291-5736 291-5753 291-5770 291-6005 292-6022 292-6037 302-6302
NMSB     12-0       99-417   146-1630 148-1704 162-2132 179-2465 186-2606 191-2726 194-3050 197-3164 221-3746 226-4072 236-4447 238-4534
        248-5023 258-5276 261-5404 297-6172 303-6345 324-7077 327-7162
NNEG     12-0       99-400    99-404   99-405  106-525  129-1152 133-1217 201-3306 201-3307 318-6705 322-7008
NODEID   42-0       91-255
NOPREG   18-0
NOVER    14-0
NPKTS    52-0
NPOLL    12-0       13-0      13-0     119-1021 133-1222
NPRDY    14-0
NRPE     14-0
NRPOK    12-0       13-0
NRRDY    14-0       14-0
NSCAN    12-0       13-0      13-0      13-0
NSCB     65-0
NSDI     65-0       65-0     124-1072 145-1603
NSECTR   14-0
NSEEKS   49-0       49-0     195-3103 240-4625
NSTOP    13-0
NTEST    13-0      115-667   115-671  120-1026 120-1030 145-1625
NUPF     12-0      192-2737 193-3000 232-4254 239-4546 323-7031
NWRC     14-0
NZRO     12-0       78-20     78-24    81-71    82-76    83-100   83-103   83-107   84-120   84-127   84-134   85-140   85-150   86-153
         87-176     87-210    87-213   88-216   88-217   88-222   88-225   90-244   90-247   95-326   95-327   95-330   95-332   95-335
```

```
          96-341   96-342   98-363   99-376   99-410  100-442  100-443  101-450  101-454  102-463  103-473  104-477  106-514  106-517
         106-523  107-534  108-545  108-603  109-606  110-625  113-640  113-642  114-651  114-653  114-663  114-665  115-702  116-737
         117-753  118-1012 120-1047 122-1055 122-1061 122-1066 131-1200 136-1240 136-1244 136-1251 137-1260 138-1306 140-1525 142-1542
         142-1543 142-1551 142-1561 143-1563 143-1566 144-1601 145-1607 145-1612 145-1615 147-1643 147-1645 147-1653 148-1700 149-1714
         149-1726 149-1731 150-1741 151-1770 155-2021 156-2046 160-2070 161-2077 161-2110 162-2115 162-2120 162-2140 163-2161 164-2172
         164-2173 164-2176 165-2224 166-2235 166-2236 167-2243 167-2246 167-2251 170-2275 170-2277 170-2305 170-2307 171-2311 172-2313
         172-2326 174-2350 174-2352 174-2356 174-2357 174-2360 174-2362 175-2372 176-2402 176-2410 176-2412 176-2415 177-2425 183-2516
         183-2520 183-2530 184-2545 185-2560 185-2564 185-2570 188-2621 188-2637 188-2643 191-2677 191-2701 191-2703 191-2706 191-2710
         191-2712 191-2714 191-2722 191-2731 193-2774 193-2777 193-3004 194-3043 194-3061 194-3064 195-3065 195-3105 196-3120 196-3121
         196-3122 196-3127 196-3140 196-3142 197-3150 199-3225 200-3234 201-3253 201-3264 201-3311 202-3327 206-3407 206-3411 207-3413
         207-3416 207-3420 208-3427 208-3431 208-3433 208-3435 208-3437 209-3463 209-3467 209-3477 209-3502 210-3504 210-3510 211-3557
         211-3567 211-3571 211-3601 212-3621 212-3625 214-3654 214-3656 214-3660 219-3721 220-3731 221-3737 221-3740 221-3747 221-3751
         223-3767 223-3770 223-3771 223-3772 224-3775 224-4000 224-4003 224-4006 225-4042 225-4045 226-4061 226-4075 226-4100 226-4102
         227-4106 227-4107 227-4111 227-4113 227-4122 228-4142 229-4165 230-4177 230-4212 230-4215 230-4216 230-4221 231-4226 232-4245
         232-4253 232-4262 232-4275 233-4304 233-4305 234-4347 234-4351 234-4353 234-4364 235-4410 235-4413 235-4414 236-4436 236-4437
         236-4451 237-4456 237-4474 237-4477 238-4517 238-4520 238-4524 239-4543 240-4602 240-4612 240-4620 240-4621 241-4647 242-4660
         243-4670 243-4676 243-4705 244-4711 244-4712 244-4723 244-4724 245-4734 245-4735 245-4740 245-4742 246-4753 246-4762 246-5001
         248-5020 248-5026 248-5036 250-5110 251-5134 252-5171 259-5315 259-5317 259-5356 262-5420 264-5440 264-5441 267-5455 273-5510
         273-5512 280-5556 289-5650 289-5653 294-6111 295-6135 295-6140 296-6166 301-6255 301-6260 304-6405 306-6445 309-6511 309-6512
         311-6527 312-6561 313-6600 315-6612 315-6617 315-6621 318-6727 319-6746 319-6756 323-7032 323-7043 324-7070 324-7074 324-7117
         327-7161
OD        73-0
ODDP      22-0
ODTAF     45-0
ODTCA     45-0     221-3754
ODTCY     45-0
OFFMSK    45-0     257-5241
OFFSET    45-0     221-3743 221-3745
OFFTRK    60-0     161-2104 161-2105 246-4766
ONLINE    69-0
ONLREC    45-0     211-3575 211-3577
OP.ABO    53-0     165-2215
OP.ACC    53-0     164-2202 193-3003 226-4054 226-4074 226-4105 227-4126 235-4405 248-5017 251-5153
OP.ACP    58-0
OP.ATT    53-0     142-1551
OP.AVA    58-0     200-3252
OP.AVL    53-0     165-2210
OP.CCD    53-0     165-2207
OP.CMP    53-0     162-2117 164-2201 196-3114 196-3127 196-3143 196-3144 226-4054 226-4074 226-4105 227-4126 235-4404 241-4650 248-5017
          251-5153
OP.DAP    53-0     165-2213 174-2356
OP.DUP    58-0
OP.END    58-0     147-1645 177-2427 182-2506
OP.ERS    53-0     164-2203 193-3003 226-4054 226-4074 226-4105 227-4126 248-5017 251-5153
OP.FLU    53-0     165-2206
OP.GCS    53-0     165-2216
OP.GST    53-0     172-2313
OP.GUS    53-0     165-2217
OP.MRD    53-0     165-2226
OP.MWR    53-0     165-2227
OP.ONL    53-0     165-2211 226-4054 226-4074 226-4075 226-4105 227-4126 234-4327 248-5017 248-5020 251-5153
OP.RD     53-0     162-2117 164-2177 196-3117 196-3122 196-3122 196-3122 196-3122 196-3144 226-4054 226-4074 226-4105 227-4106 227-4126
          241-4650 248-5017 251-5153
OP.RPL    53-0     165-2204 193-3003 226-4054 226-4074 226-4105 227-4126 248-5017 251-5153
```



```
OP.SCC    53-0     165-2220
OP.SUC    53-0     165-2212 209-3470 210-3514 211-3547
OP.WR     53-0     162-2117 164-2200 196-3120 196-3122 196-3122 196-3122 196-3122 196-3144 226-4054 226-4074 226-4105 227-4126 230-4206
          241-4650 248-5017 251-5153
OPCOD     46-0
OPCODE    49-0      49-0    225-4034 226-4053 226-4072 226-4104 227-4125 248-5016 249-5074 251-5152
OPM       22-0
OVER      14-0
OVLPER   172-2327 173-2330
OW        44-0     289-5655 293-6077 295-6121 296-6143 296-6166 297-6177 301-6262 306-6431
OWN       35-0     146-1630 147-1662 148-1704 149-1723 151-1765 151-1774
P         52-0      52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0     52-0
          52-0      52-0     52-0     52-0     52-0     52-0     52-0     52-0     53-0     53-0     53-0     54-0     54-0     54-0
          54-0      54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0
          54-0      54-0     54-0     54-0     54-0     54-0     54-0     54-0     54-0     55-0     55-0     55-0     55-0     55-0
          55-0      55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0     55-0
          55-0      55-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0
          57-0      57-0     57-0     57-0    147-1652 149-1726 149-1731 154-2011 165-2232 167-2243 167-2246 167-2251 170-2277 170-2305
         174-2352 183-2520 184-2542 188-2621 191-2701 191-2710 191-2722 191-2731 192-2742 192-2755 193-3017 198-3206 200-3234 208-3431
         209-3467 214-3654 214-3660
P.BCNO    54-0      54-0    182-2503 182-2511 188-2632 191-2731 192-2732 192-2755 193-3000 197-3165 198-3210 198-3210 204-3377 257-5253
         264-5432 333-7731
P.BCN1    54-0     182-2504 182-2512 188-2633 191-2727 192-2733 193-3000 193-3000 197-3165 197-3165 198-3210 257-5254
P.BUFO    54-0      54-0     54-0    167-2247 167-2251 197-3177 197-3200 257-5237
P.BUF1    54-0      54-0     54-0    191-2725 226-4071 248-5022 257-5243
P.BUF2    54-0      54-0    221-3752 257-5245
P.BUF3    54-0      54-0    257-5247
P.BUF4    54-0      54-0     54-0
P.BUF5    54-0      54-0    197-3200
P.BUFH    54-0     193-3005 193-3005 204-3376
P.BUFL    54-0     193-3005 204-3375
P.CMSO    55-0      55-0    183-2533
P.CMS1    55-0     183-2531
P.CNTO    55-0      55-0    214-3667
P.CNT1    55-0      55-0
P.CNT2    55-0      55-0
P.CNT3    55-0
P.CNTF    54-0     214-3670
P.CONS    54-0
P.CRFO    53-0      53-0     62-0    184-2544 202-3330 257-5251
P.CRF1    53-0      62-0    184-2545 202-3331 257-5252
P.CSVR    55-0     214-3666
P.CTMO    55-0     214-3664 214-3666
P.CYL     55-0
P.CYLS    54-0     189-2647 189-2650
P.ELGO    54-0      54-0
P.ELG1    54-0
P.FBBO    55-0      55-0    197-3175 197-3175
P.FBB1    55-0     197-3175
P.FLGS    55-0     202-3332 240-4605 240-4615
P.GRP     55-0     189-2646 189-2647
P.HSTO    54-0      54-0
P.HST1    54-0
P.HTMO    54-0     214-3661
```

```
P.LBNO   54-0      54-0     191-2720 191-2722 192-2757 193-3017 194-3034 197-3156 210-3523 210-3527 221-3760
P.LBN1   54-0     191-2720 191-2720 192-2757 192-2757 194-3034 194-3034 197-3156 197-3156 210-3524 210-3530 221-3761
P.LCKA  269-5472 269-5473
P.LINK   53-0      53-0
P.LOCK  138-1312 145-1605 209-3441 232-4260 237-4465 269-5473
P.MED1   55-0      55-0     217-3705 217-3706
P.MED2   55-0     217-3706 217-3707
P.MLUN   55-0     217-3702 217-3703
P.MOD    54-0      62-0     162-2113 169-2264 170-2277 170-2305 174-2352 175-2370 183-2520 188-2621 188-2622 191-2701 191-2710 195-3112
        208-3431 209-3467 211-3560 214-3654 227-4115 233-4321 238-4540
P.OPCD   54-0      62-0     142-1547 146-1641 149-1727 149-1731 162-2112 164-2166 165-2231 165-2232 173-2342 177-2426 182-2507 195-3113
        196-3125 225-4033 227-4124 239-4556 241-4647
P.RBNO   54-0      54-0     192-2742
P.RBN1   54-0
P.RBNS   55-0     189-2657 212-3637
P.RCTC   55-0
P.RCTS   55-0     189-2652 189-2657 212-3636
P.RFNO   54-0      54-0     184-2536
P.RFN1   54-0     184-2537
P.RGI0   54-0      54-0     165-2225
P.RGI1   54-0
P.RGO0   54-0      54-0     204-3372
P.RGO1   54-0
P.RSO3   54-0      62-0     149-1725 149-1726 183-2525 195-3111 196-3130 197-3202 233-4276 239-4551 239-4554
P.RSO6   54-0     167-2241 167-2243 197-3201 198-3206 211-3574 211-3604
P.RSO8   54-0
P.RS16   54-0     167-2244 167-2246 188-2634
P.RS17   54-0     188-2635
P.RS19   55-0
P.SHST   55-0     212-3632 217-3710
P.SHUN   54-0     217-3707 217-3710
P.STS    55-0     176-2420
P.TRCK   55-0     189-2644 189-2646
P.UNFL   54-0     210-3506 217-3703
P.UNIT   54-0      62-0     185-2551 187-2610 188-2623 188-2631 200-3251 201-3257
P.UNSO   55-0      55-0     212-3632 212-3633
P.UNS1   55-0     212-3633 212-3634
P.UNTO   55-0      55-0     216-3676 217-3704
P.UNT1   55-0      55-0     216-3676 216-3677
P.UNT2   55-0      55-0     216-3677
P.UNT3   55-0     217-3705
P.USVR   55-0     189-2650 189-2652
P.VCID   53-0     147-1671 149-1732 163-2163 164-2170 176-2421 184-2547 201-3267 211-3555
P.VRSN   54-0     214-3657 214-3660
P.VSEO   55-0      55-0     212-3634 212-3635
P.VSE1   55-0     212-3635
PABRT    53-0     163-2164 170-2302
PACTV    53-0     173-2332 174-2347 191-2676 191-2705 199-3231 208-3426 241-4646
PAGESZ   46-0     257-5232 257-5255 257-5255 257-5261
PAR      18-0      18-0      18-0     101-447  101-453  139-1314 219-3724 228-4146 258-5311 261-5406 289-5665
PARTST  101-445
PKIP     59-0     142-1555 163-2162 196-3132 197-3205
PKT001   52-0      52-0
PKT002   52-0      52-0
```



```
PKT003   52-0      52-0
PKT004   52-0      52-0
PKT005   52-0      52-0
PKT006   52-0      52-0
PKT007   52-0      52-0
PKT008   52-0      52-0
PKT009   52-0      52-0
PKT010   52-0      52-0
PKT011   52-0      52-0
PKT012   52-0      52-0
PKT013   52-0      52-0
PKT014   52-0      52-0
PKT015   52-0      52-0
PKT016   52-0      52-0
PKT017   52-0      52-0
PKT018   52-0      52-0
PKT019   52-0      52-0
PKT020   52-0      52-0     164-2171 202-3326
PKTBUF   52-0      52-0     154-2007 184-2550
PKTEND   52-0      65-0     154-2005 154-2005 154-2006 184-2540 184-2541
PLOCK    19-0     138-1304 138-1305 138-1311 138-1313 140-1516 145-1613 173-2341 230-4172 233-4305 269-5472 269-5474
POLL     12-0      12-0      13-0     119-1024 134-1225 142-1537
POLTST   86-173   119-1014
POLY    306-6450 308-6473 316-6630 319-6744
POLYP   306-6450 319-6745 319-6750
POLYX   319-6744 319-6746
PONER    53-0     211-3556 211-3557
POP1    329-7424
POP2    258-5311 329-7425
PRDY     14-0
PSTACK   78-25     82-76
PSTAT    53-0     149-1707 154-2012 172-2321 173-2332 173-2335 184-2544 199-3227 199-3231 241-4646
PTELEN   46-0     249-5050
Q        17-0      91-254    91-260    91-264    99-375    99-400    99-401    99-402    99-403    99-404    99-405    99-406    99-414    99-417
        99-420   100-442  126-1115 126-1127 126-1131 126-1133 128-1137 128-1141 128-1143 129-1151 129-1152 130-1165 131-1170 131-1172
       131-1174 131-1175 131-1177 132-1211 132-1213 133-1214 138-1274 138-1275 138-1300 138-1307 138-1310 139-1314 139-1315 139-1316
       140-1513 140-1523 142-1557 142-1560 145-1604 145-1605 145-1607 145-1617 145-1620 145-1621 146-1637 147-1662 148-1701 148-1702
       149-1710 149-1723 153-2004 161-2101 161-2111 162-2144 162-2147 162-2150 163-2152 166-2235 167-2245 167-2250 167-2253 168-2254
       168-2255 168-2256 168-2260 168-2261 168-2262 169-2263 169-2265 169-2267 170-2270 170-2271 170-2272 170-2273 170-2276 174-2344
       174-2345 174-2346 176-2404 176-2411 176-2412 179-2435 179-2462 179-2464 179-2465 180-2466 181-2500 181-2501 183-2517 185-2554
       185-2563 185-2571 186-2576 186-2600 187-2612 187-2613 188-2617 188-2620 188-2642 189-2644 190-2662 190-2663 190-2664 190-2665
       190-2666 190-2667 190-2670 190-2671 191-2672 191-2673 191-2674 191-2675 191-2720 191-2730 192-2734 192-2735 192-2745 192-2750
       192-2752 192-2757 193-2773 193-3000 193-3002 193-3005 193-3010 194-3034 194-3035 194-3036 194-3040 194-3047 195-3101 197-3156
       197-3165 197-3173 197-3175 198-3210 198-3212 198-3214 200-3244 201-3274 201-3314 202-3317 202-3325 202-3327 202-3334 202-3341
       202-3346 203-3354 203-3370 204-3373 204-3374 208-3422 208-3423 208-3424 208-3425 209-3444 209-3446 209-3471 210-3516 210-3526
       210-3541 211-3576 212-3610 212-3623 212-3626 214-3653 214-3664 214-3667 217-3702 217-3704 221-3741 221-3755 223-3773 226-4052
       227-4127 227-4135 230-4172 230-4173 232-4272 233-4311 233-4312 234-4347 234-4351 234-4353 234-4357 235-4400 235-4403 235-4426
       236-4435 236-4436 237-4454 237-4455 240-4600 240-4602 240-4616 240-4617 240-4623 240-4627 241-4630 241-4646 241-4650 242-4656
       243-4676 243-4700 244-4722 245-4737 246-4745 246-4746 246-4754 246-4762 246-4763 246-4764 246-4770 246-4775 246-4777 246-5001
       247-5005 247-5007 249-5066 249-5067 250-5103 250-5105 250-5110 251-5130 251-5131 251-5137 251-5145 251-5161 251-5162 251-5163
       251-5164 252-5167 252-5170 252-5172 257-5261 257-5262 257-5271 258-5305 258-5307 259-5324 259-5326 259-5331 259-5336 259-5344
       259-5352 262-5420 264-5437 272-5501 272-5503 272-5504 273-5507 273-5511 273-5512 280-5547 280-5555 281-5560 281-5561 281-5565
       282-5572 283-5613 289-5663 289-5665 293-6075 293-6077 293-6100 294-6112 295-6123 295-6124 295-6141 299-6230 299-6234 303-6327
       303-6351 303-6352 303-6360 304-6407 304-6410 304-6411 304-6412 305-6423 305-6424 306-6446 308-6460 308-6463 308-6470 309-6504
```

LSCS FORM=QUAD

```
        309-6506 312-6553 315-6605 317-6656 317-6657 318-6707 318-6710 318-6711 318-6712 318-6714 318-6717 318-6720 318-6721 318-6723
        318-6725 319-6741 319-6742 322-6765 322-7001 322-7003 323-7054 324-7062 324-7066 324-7067 324-7070 324-7074 324-7126 325-7130
        325-7131 325-7132 325-7134 326-7147 326-7150 331-7441 331-7442 331-7443 333-7731
Q.STAT   46-0     162-2146 172-2320 173-2327 199-3224 218-3716
QB       73-0
QB.RD   289-5634 289-5635
QB.WR   301-6242 301-6242
QDADM     1-0
QREVB     1-0
QUESAV   49-0     49-0     147-1651 147-1654
QW       44-0
RO       81-62    81-63    81-64    81-70    81-70    81-71    83-77    83-77    83-102   83-106   83-107   84-114   84-116   84-120
         86-163   87-177   87-205   87-206   87-210   87-211   88-220   90-243   90-244   90-246   90-247   91-253   91-256   91-262
         92-256   92-267   92-272   92-273   92-274   93-303   93-304   94-314   94-315   95-331   95-334   97-352   98-356   98-367
         98-370   98-372   99-373   99-374   99-376   99-377   99-406   99-407   99-411   99-413   99-415   99-416   100-421  100-422
        100-441  100-443  106-507  106-513  106-522  107-537  108-576  110-623  112-630  112-631  112-633  113-634  113-635  113-636
        113-636  113-640  114-644  114-644  114-651  114-654  114-655  114-657  114-660  114-661  114-661  114-663  115-672  115-673
        115-675  115-676  115-700  115-700  115-701  115-714  116-720  116-721  116-725  116-726  116-735  116-735  116-736  116-744
        117-756  117-765  117-766  117-767  118-776  118-777  118-1000 118-1006 118-1010 119-1014 119-1017 119-1020 120-1032 120-1037
        120-1042 120-1043 122-1055 122-1057 122-1061 122-1064 125-1104 126-1122 126-1125 126-1126 128-1145 129-1147 129-1150 129-1155
        130-1160 130-1163 132-1205 132-1207 136-1240 136-1242 136-1244 136-1247 137-1254 137-1256 137-1260 137-1263 137-1271 137-1272
        140-1521 140-1523 140-1525 140-1527 142-1550 144-1600 145-1615 146-1635 147-1643 147-1650 147-1652 149-1713 149-1717 149-1730
        149-1733 150-1741 150-1743 150-1745 150-1750 150-1753 150-1753 150-1754 151-1757 151-1760 151-1761 151-1763 151-1767 151-1773
        151-1774 151-1776 152-2002 154-2012 155-2016 155-2017 156-2030 156-2033 156-2037 156-2041 156-2047 158-2051 158-2052 161-2104
        161-2105 161-2106 161-2107 161-2110 162-2114 162-2117 162-2125 162-2133 162-2134 162-2140 162-2143 163-2165 164-2177 164-2200
        164-2201 164-2202 164-2203 165-2204 165-2206 165-2207 165-2210 165-2211 165-2212 165-2213 165-2215 165-2216 165-2217 165-2220
        165-2226 165-2227 170-2303 173-2337 173-2340 174-2350 174-2380 175-2367 175-2372 175-2374 175-2376 176-2377 176-2400 176-2401
        176-2412 176-2415 176-2417 176-2422 176-2423 179-2450 179-2453 179-2455 179-2457 179-2462 180-2476 182-2505 182-2506 182-2514
        183-2527 185-2555 185-2557 185-2564 185-2565 185-2567 185-2570 186-2574 186-2575 186-2601 186-2603 186-2604 186-2606 186-2607
        187-2613 187-2614 187-2616 188-2637 188-2641 189-2651 189-2654 189-2657 189-2660 191-2717 191-2723 191-2724 192-2735 192-2737
        192-2741 192-2746 193-2770 193-2771 193-2774 193-2776 193-3001 193-3006 193-3015 194-3042 194-3043 194-3050 195-3077 195-3101
        195-3107 195-3110 196-3114 196-3117 196-3120 196-3122 196-3123 196-3143 197-3147 197-3157 197-3162 197-3167 197-3170
        197-3174 197-3200 198-3206 200-3234 200-3243 200-3252 201-3255 201-3256 201-3260 201-3263 201-3264 201-3271 201-3272 201-3273
        201-3305 201-3306 201-3310 201-3311 202-3316 202-3323 202-3333 202-3335 202-3344 203-3350 203-3360 203-3361 203-3362 203-3363
        203-3365 203-3366 203-3367 204-3400 205-3401 205-3402 205-3402 205-3403 205-3403 205-3404 205-3405 206-3407 206-3412 207-3413
        207-3420 207-3421 208-3437 209-3451 209-3455 209-3465 209-3472 209-3477 209-3502 209-3503 210-3504 210-3511 210-3515 210-3525
        210-3533 211-3562 211-3567 211-3570 212-3606 212-3623 213-3650 213-3651 214-3665 214-3666 214-3671 215-3674 216-3701 217-3707
        217-3710 219-3721 219-3722 219-3723 219-3724 220-3725 220-3726 220-3727 220-3731 220-3732 220-3736 221-3745 221-3746 221-3747
        221-3751 221-3756 221-3757 222-3764 224-4000 224-4002 224-4004 225-4017 225-4037 225-4042 225-4043 225-4045 225-4051 226-4055
        226-4061 226-4073 226-4074 226-4075 227-4107 227-4111 227-4131 227-4135 229-4156 229-4157 229-4162 229-4163 230-4166 230-4211
        230-4213 230-4214 230-4215 230-4215 231-4224 231-4230 231-4231 232-4244 232-4256 232-4260 232-4261 232-4267 232-4270 232-4275
        233-4277 233-4303 233-4304 233-4306 233-4316 234-4334 234-4341 234-4343 234-4344 234-4346 234-4350 234-4352 234-4361 234-4363
        235-4370 235-4376 235-4377 235-4407 235-4411 235-4412 235-4413 235-4413 235-4415 235-4425 236-4430 236-4443 237-4461 237-4461
        237-4462 237-4476 237-4500 237-4503 237-4503 237-4510 237-4510 237-4511 237-4511 238-4513 238-4514 238-4516 238-4517 239-4544
        239-4545 239-4546 239-4562 239-4563 239-4576 240-4606 240-4607 240-4610 241-4634 243-4667 243-4670 243-4673 243-4702 243-4707
        244-4711 244-4711 244-4715 244-4723 244-4726 245-4733 245-4734 245-4741 245-4742 246-4745 246-4752 246-4754 246-4761 246-4766
        246-5002 246-5003 247-5006 247-5007 247-5012 247-5013 248-5017 248-5020 248-5030 248-5032 249-5053 249-5056 249-5057 249-5057
        249-5060 249-5080 249-5061 249-5061 249-5062 249-5100 253-5202 253-5203 253-5203 253-5204 253-5204 253-5205 253-5205 253-5206
        253-5207 253-5210 253-5217 253-5222 254-5223 254-5223 255-5227 257-5232 257-5260 257-5262 257-5263 257-5264 257-5266 257-5271
        258-5274 258-5275 258-5276 258-5277 258-5277 258-5300 258-5300 258-5301 258-5301 258-5302 258-5302 258-5303 258-5303 258-5307
        259-5322 259-5325 259-5331 259-5333 259-5342 259-5346 259-5347 259-5350 259-5363 259-5370 259-5377 261-5402 261-5405 261-5406
        262-5412 262-5413 262-5414 264-5441 264-5442 267-5456 268-5462 268-5485 271-5477 274-5524 274-5525 275-5530 275-5531 276-5532
        276-5533 279-5545 280-5546 280-5550 280-5550 280-5551 280-5552 280-5556 280-5556 280-5557 282-5573 282-5603 282-5604 282-5605
        282-5610 283-5614 283-5615 283-5616 289-5644 289-5660 293-6064 293-6066 294-6110 294-6110 295-6117 295-6120 295-6130 295-6137
```



```
        296-6151 296-6167 297-6171 299-6214 299-6217 299-6220 299-6221 299-6223 299-6224 299-6225 301-6251 301-6260 301-6267 302-6305
        303-6326 303-6330 303-6344 303-6345 303-6357 304-6405 304-6413 305-6421 306-6445 307-6453 308-6461 308-6464 308-6473 309-6500
        309-6503 309-6504 309-6504 309-6512 312-6536 312-6537 312-6544 312-6545 313-6563 313-6565 313-6566 315-6610 315-6613 315-6613
        315-6617 315-6617 315-6627 316-6633 316-6634 317-6646 317-6660 317-6660 318-6676 318-6722 319-6750 319-6756 319-6756 323-7032
        323-7036 323-7040 323-7051 323-7052 324-7061 324-7066 324-7076 324-7077 324-7100 324-7115 324-7116 324-7117 324-7121 325-7137
        327-7155 327-7161 331-7444 333-7733 333-7745 333-7746 334-7751 334-7752 334-7753 334-7754 334-7755 334-7756 334-7757
R1       81-65    81-67    84-133   86-164   87-203   87-204   87-207   87-213   87-214   91-255   91-257   91-261   93-306   94-321
         95-323   95-325   95-332   96-340   96-342   97-351   98-355   98-356   98-360   98-362   98-363   99-376   99-377   99-402
         99-410   99-412   100-422  100-423  102-456  102-457  102-462  102-463  102-464  102-465  106-514  106-516  106-522  106-523
        106-526  115-706  116-723  116-730  119-1015 119-1016 120-1031 120-1033 120-1036 120-1041 122-1054 122-1060 126-1117 129-1156
        131-1176 131-1177 131-1200 131-1201 133-1220 136-1237 136-1243 137-1255 137-1257 137-1261 137-1264 137-1265 137-1266 137-1267
        137-1270 138-1306 138-1310 139-1315 139-1321 139-1322 140-1506 140-1507 140-1515 140-1517 140-1522 142-1545 142-1546 142-1552
        142-1555 144-1572 144-1573 144-1575 144-1576 146-1633 146-1634 147-1656 147-1666 147-1667 147-1670 147-1672 147-1673 149-1720
        149-1726 149-1731 150-1741 150-1743 151-1757 152-2001 153-2003 155-2013 156-2031 158-2054 160-2074 162-2126 162-2130 162-2132
        162-2135 162-2136 162-2142 165-2232 166-2233 166-2236 166-2237 167-2241 167-2242 167-2243 167-2244 167-2246 167-2247 167-2251
        169-2263 169-2265 170-2275 170-2277 170-2305 170-2307 172-2320 172-2326 173-2327 173-2331 174-2347 174-2352 174-2355 174-2357
        175-2371 176-2407 176-2414 177-2427 179-2444 179-2450 179-2451 179-2454 179-2455 182-2502 182-2505 182-2513 182-2514 183-2516
        183-2520 184-2546 187-2614 188-2621 189-2656 189-2657 191-2676 191-2701 191-2705 191-2710 191-2712 191-2713 191-2716 191-2722
        191-2731 192-2742 192-2752 192-2755 193-2777 193-3017 193-3030 194-3055 194-3063 195-3105 197-3157 197-3167 197-3170 197-3176
        197-3177 197-3201 197-3205 198-3207 198-3213 198-3220 199-3222 199-3223 199-3224 199-3225 200-3235 200-3250 201-3262 208-3426
        208-3431 208-3433 209-3466 209-3467 209-3476 209-3501 209-3502 210-3507 210-3513 210-3515 210-3517 210-3520 210-3525 211-3553
        211-3556 211-3557 211-3566 211-3575 211-3577 212-3615 212-3625 214-3654 214-3656 214-3660 215-3673 215-3674 221-3742 221-3744
        221-3750 223-3766 223-3767 223-3770 223-3771 223-3772 224-4005 224-4006 225-4013 225-4014 225-4025 225-4026 225-4035 226-4060
        226-4105 227-4106 227-4121 227-4123 227-4124 227-4126 227-4127 227-4130 227-4131 227-4132 227-4132 227-4133 227-4133 227-4134
        227-4134 227-4136 227-4136 228-4143 228-4144 228-4146 228-4146 230-4170 230-4210 230-4217 230-4220 233-4322 234-4327 234-4336
        234-4354 234-4362 235-4374 235-4376 235-4377 235-4401 235-4406 235-4422 236-4447 236-4450 237-4504 237-4507 238-4515 238-4522
        238-4523 238-4531 238-4532 238-4534 238-4541 239-4560 239-4561 241-4636 242-4657 242-4657 244-4715 244-4720 244-4720 246-4750
        246-4751 246-4753 246-4772 246-5000 248-5025 248-5032 249-5050 249-5053 249-5055 249-5056 253-5177 253-5206 253-5210 255-5226
        255-5227 257-5241 257-5260 257-5266 258-5304 258-5304 258-5305 258-5306 258-5310 258-5310 259-5316 259-5332 259-5335 259-5336
        259-5344 259-5371 264-5431 264-5433 264-5434 264-5443 264-5445 267-5457 272-5500 278-5537 278-5540 278-5541 279-5544 279-5544
        280-5553 280-5553 280-5554 280-5554 282-5574 283-5617 283-5620 289-5635 290-5674 290-5675 290-5701 290-5702 290-5710 290-5711
        290-5715 290-5717 290-5725 290-5726 290-5732 291-5734 291-5742 291-5743 291-5747 291-5751 291-5757 291-5760 291-5764 291-5766
        291-5774 291-5775 291-6001 291-6003 292-6011 292-6012 292-6016 292-6020 292-6026 292-6027 292-6033 292-6035 293-6042 293-6047
        293-6050 293-6051 293-6053 293-6054 293-6060 293-6067 293-6070 293-6071 293-6072 293-6073 293-6074 294-6106 295-6132 296-6155
        299-6215 299-6226 301-6242 301-6271 301-6272 301-6274 301-6277 301-6300 302-6313 302-6315 302-6317 302-6321 302-6323 303-6336
        303-6340 303-6353 304-6376 304-6400 304-6414 306-6435 306-6436 306-6440 308-6457 308-6461 308-6462 308-6464 308-6466 308-6471
        309-6505 309-6511 311-6520 311-6526 312-6536 312-6544 313-6577 313-6600 315-6625 316-6631 316-6640 316-6641 318-6677 318-6702
        318-6702 319-6747 319-6747 322-6763 322-6771 322-7013 322-7017 322-7021 323-7042 323-7044 324-7075 324-7102 324-7125 324-7125
        325-7131 325-7132 325-7134 326-7143 326-7145 326-7146 327-7160 331-7440 331-7444 333-7725
R10      17-0     17-0     100-431  100-432  106-510  106-525  106-527  108-542  108-544  108-577  108-602  110-621  110-626  117-763
        118-1013 122-1055 122-1061 122-1067 124-1103 125-1104 131-1166 131-1201 132-1205 136-1240 136-1244 136-1252 137-1256 146-1641
        147-1644 147-1647 147-1671 149-1711 149-1715 149-1725 149-1727 149-1732 154-2005 154-2006 154-2007 154-2010 154-2011 162-2112
        162-2113 162-2122 162-2127 162-2145 162-2146 163-2151 163-2163 164-2166 164-2167 164-2170 164-2171 164-2174 165-2221 165-2225
        165-2231 166-2233 167-2242 167-2252 169-2264 169-2266 172-2315 172-2316 172-2321 173-2332 173-2333 173-2334 175-2370 176-2420
        176-2421 177-2426 179-2436 179-2446 182-2503 182-2504 182-2507 182-2511 182-2512 183-2525 183-2531 183-2533 184-2536 184-2537
        185-2551 187-2610 188-2622 188-2623 188-2631 188-2632 188-2633 188-2634 188-2635 189-2644 191-2720 191-2725 191-2727 192-2732
        192-2733 192-2757 192-2765 192-2767 193-3000 193-3005 193-3010 193-3011 193-3012 193-3026 194-3034 194-3035 195-3067 195-3100
        195-3111 195-3112 195-3113 196-3125 196-3130 197-3154 197-3155 197-3156 197-3165 197-3175 197-3176 197-3202 198-3207 198-3210
        199-3225 199-3226 199-3232 200-3233 200-3235 200-3240 200-3242 200-3251 201-3257 201-3267 201-3274 201-3312 201-3314 202-3324
        202-3325 202-3327 202-3334 202-3341 202-3346 203-3351 203-3354 204-3372 204-3375 204-3376 204-3377 209-3447 209-3450 210-3506
        210-3523 210-3524 210-3527 210-3530 211-3543 211-3544 211-3555 211-3560 211-3574 211-3604 212-3636 212-3637 213-3641 213-3642
        214-3657 214-3661 214-3664 214-3667 214-3670 217-3702 217-3704 218-3713 218-3714 218-3716 218-3717 221-3752 221-3760 221-3761
        225-4010 225-4016 225-4021 225-4033 226-4066 226-4067 226-4071 232-4272 233-4276 233-4321 233-4326 237-4456 238-4540 239-4551
        239-4554 239-4556 240-4605 240-4615 242-4663 242-4664 248-5022 250-5115 250-5116 251-5124 251-5125 251-5126 251-5135 251-5141
```

LSCS FORM=QUAD

```
        251-5146 251-5147 251-5150 251-5156 251-5160 252-5166 252-5171 252-5173 257-5237 257-5243 257-5245 257-5247 257-5251 257-5252
        257-5253 257-5254 264-5427 264-5432 264-5443 264-5445 282-5567 282-5570 282-5571 282-5602 283-5626 289-5641 289-5641 289-5650
        295-6113 295-6114 295-6115 295-6117 295-6120 295-6130 295-6134 299-6215 299-6226 301-6246 301-6246 312-6554 312-6555 312-6555
        315-6607 315-6615 315-6615 315-6623 315-6624 316-6636 316-6637 316-6645 317-6650 317-6671 318-6705 318-6706 318-6706 318-6707
        318-6710 318-6716 319-6751 319-6752 322-7001 324-7060 324-7060 324-7071 324-7072 324-7073 324-7110 324-7110 333-7731
R11      17-0   100-432  100-433  122-1070 160-2066 261-5403 261-5404 261-5405 262-5417 262-5420 263-5426 268-5467 268-5470 268-547
        274-5526 274-5527 282-5612 283-5627 289-5634 289-5634 333-7737 333-7740
R12      17-0    17-0    100-433  100-434  120-1045 120-1047
R13      17-0    78-24    78-25   100-434  100-435
R14      17-0   100-435  100-436
R15      17-0    17-0    100-436  100-437
R16      17-0   100-437  100-440
R17      37-0    76-0     76-2     78-17    78-20    78-21    78-22    78-23    82-76    83-110   84-123   100-430  100-431
R2       87-200  87-202   87-212   88-221   89-236   89-240   89-241   89-242   94-322   95-324   95-326   95-330   95-337   96-341
         97-350  98-353   98-361   98-364   98-371  100-423  100-424  102-460  102-461  102-462  104-503  104-504  106-511  106-520
        106-521 108-540  108-546  108-561  108-562  108-566  108-567  108-575  109-605  110-622  110-624  115-710  115-715  116-724
        116-745 117-757  117-760  117-761  118-1007 118-1011 120-1035 120-1040 122-1063 122-1065 124-1077 124-1100 124-1102 126-1123
        126-1127 128-1140 128-1144 129-1151 129-1156 130-1161 134-1230 136-1246 136-1250 140-1520 140-1524 143-1563 145-1603 145-1611
        147-1642 147-1645 148-1675 148-1676 148-1677 148-1700 148-1700 150-1754 155-2023 155-2024 155-2025 156-2032 156-2040 156-2042
        156-2043 156-2044 156-2046 161-2100 162-2121 162-2125 162-2130 164-2172 164-2173 165-2224 165-2230 172-2323 174-2350 174-2356
        174-2363 174-2364 176-2403 176-2406 176-2410 176-2411 179-2441 179-2446 179-2453 179-2465 180-2466 180-2475 180-2476 183-2523
        183-2532 184-2540 184-2541 184-2542 184-2544 184-2545 184-2547 184-2550 186-2603 188-2624 188-2626 188-2627 191-2714 191-2715
        191-2721 192-2753 192-2754 192-2764 193-2773 193-3007 193-3016 193-3031 194-3046 194-3046 194-3050 194-3053 194-3054 194-3064
        196-3121 197-3160 197-3166 197-3171 198-3211 198-3217 199-3231 200-3246 201-3261 201-3303 201-3313 202-3345 203-3357 206-3406
        209-3447 209-3450 209-3462 209-3474 210-3505 210-3510 210-3511 210-3513 210-3533 210-3541 211-3550 225-4012 225-4024 225-4036
        225-4040 225-4041 230-4206 230-4207 230-4216 231-4222 232-4257 232-4262 232-4270 233-4326 235-4404 235-4405 235-4416 236-4451
        237-4452 237-4457 237-4463 237-4464 237-4466 240-4601 240-4603 240-4626 240-4626 241-4631 241-4646 241-4647 242-4660 242-4660
        243-4672 244-4721 244-4721 246-4756 246-4757 246-4760 246-4760 246-4761 246-5000 253-5200 253-5211 253-5213 253-5216 253-5216
        253-5221 253-5221 257-5255 258-5313 259-5362 259-5363 259-5367 259-5367 259-5370 264-5433 264-5434 273-5514 273-5520 280-5551
        280-5552 282-5575 283-5621 289-5644 289-5645 289-5646 289-5657 290-5672 290-5677 290-5704 290-5706 290-5713 290-5721 290-5723
        290-5730 291-5736 291-5740 291-5745 291-5753 291-5755 291-5762 291-5770 291-5772 291-5777 291-6005 292-6007 292-6014 292-6022
        292-6024 292-6031 292-6037 293-6062 296-6165 299-6216 299-6227 301-6251 301-6252 301-6253 301-6275 302-6303 304-6367 304-6370
        304-6404 306-6444 308-6455 308-6467 308-6474 308-6475 308-6476 311-6521 311-6523 311-6524 311-6525 311-6525 312-6547 312-6552
        312-6553 315-6602 315-6603 315-6605 315-6606 315-6621 317-6655 318-6700 318-6713 318-6713 318-6740 319-6761 319-6761 322-6772
        322-6772 322-6777 323-7024 323-7025 323-7026 323-7027 323-7031 323-7035 324-7065 324-7073 325-7132 325-7133
R3       17-0    77-5    100-424  100-425  124-1072 124-1076 126-1111 126-1113 126-1116 126-1121 126-1130 128-1142 128-1144 138-1277
        138-1302 143-1566 143-1567 143-1571 147-1665 147-1666 147-1667 147-1670 147-1673 161-2111 162-2144 162-2145 162-2147 170-2301
        170-2302 179-2433 179-2437 179-2461 180-2470 180-2473 180-2474 184-2543 186-2575 186-2602 187-2615 192-2766 196-3133 210-3520
        211-3551 211-3552 211-3553 211-3572 212-3607 212-3617 212-3620 212-3621 212-3624 212-3627 212-3630 212-3631 224-3774 230-4176
        230-4202 230-4205 230-4217 230-4221 230-4221 231-4223 232-4247 232-4247 232-4252 232-4252 232-4253 232-4254 232-4273 232-4274
        234-4333 235-4422 235-4423 235-4424 236-4443 237-4504 237-4507 239-4562 239-4576 240-4611 240-4612 240-4620 240-4621 240-4622
        241-4643 243-4671 243-4674 243-4675 250-5105 250-5106 250-5110 250-5120 250-5120 251-5131 251-5133 251-5144 253-5214 253-5220
        258-5312 263-5426 266-5451 282-5576 283-5622 289-5655 289-5664 289-5671 293-6077 295-6121 295-6136 296-6143 296-6145 296-6152
        296-6166 297-6175 297-6176 301-6262 301-6270 302-6310 303-6334 303-6343 303-6351 304-6365 304-6374 304-6403 304-6411 305-6426
        306-6431 306-6442 307-6453 307-6454 309-6477 309-6501 309-6507 309-6507 312-6541 312-6545 312-6556 312-6556 312-6560 313-6563
        315-6627 316-6634 317-6662 317-6670 317-6674 318-6726 318-6727 318-6733 318-6734 319-6757 319-6760 319-6760 323-7034 323-7037
        324-7070 324-7074 324-7126 325-7134 331-7441 331-7443 333-7744
R4       17-0    17-0    100-425  100-426
R5       17-0   100-426  100-427  127-1134 131-1173 132-1202 132-1204 133-1214 137-1262 137-1272 138-1303 283-5623 308-6476 312-6551
        312-6552 312-6557 312-6561 313-6565 315-6611 315-6612 315-6614 315-6616 315-6620 316-6630 316-6631 316-6642 316-6643 317-6651
        317-6653 317-6654 317-6661 317-6663 318-6717 318-6723 318-6735 318-6736 318-6737 318-6740 319-6741 319-6741 319-6742 319-6743
        319-6744 319-6745 319-6746 319-6752 319-6754 319-6755
R6       17-0    86-154   86-155   86-156   86-170   86-171   88-217   88-224  100-427  100-430  106-505  106-517  106-524  107-531
        107-532 108-552  108-553  113-642  114-666  115-712  115-713  116-742  116-743  117-754  117-755  119-1022 119-1023 162-2140
```



```
        162-2143 163-2157 163-2161 163-2162 163-2164 173-2336 173-2343 175-2373 175-2376 179-2434 179-2440 179-2444 179-2451 179-2462
        179-2464 186-2572 186-2602 186-2605 186-2605 188-2640 188-2643 189-2660 193-3015 193-3026 194-3054 194-3060 194-3061 194-3063
        195-3065 195-3070 195-3071 195-3073 195-3104 195-3107 196-3115 196-3123 196-3131 196-3132 196-3144 196-3145 197-3160 197-3161
        197-3163 197-3163 197-3164 197-3164 197-3166 197-3171 198-3220 200-3245 202-3326 202-3330 202-3331 202-3332 202-3336 209-3464
        210-3540 211-3563 211-3601 211-3602 213-3641 213-3642 213-3647 224-3775 225-4012 225-4013 225-4017 225-4022 225-4050 227-4114
        227-4115 229-4152 229-4160 229-4164 230-4204 231-4233 232-4241 232-4245 232-4245 232-4246 232-4251 234-4340 236-4441 236-4442
        237-4471 237-4472 237-4474 237-4477 237-4501 239-4552 239-4555 239-4557 239-4561 239-4565 239-4576 239-4575 241-4641 241-4642
        241-4644 241-4645 247-5010 247-5011 250-5114 250-5122 250-5123 250-5123 251-5133 251-5134 251-5142 251-5144 251-5153 251-5162
        252-5170 252-5174 266-5447 268-5463 268-5464 281-5563 281-5563 281-5564 281-5564 281-5566 282-5606 282-5607 282-5611 308-6456
        308-6485 311-6522 311-6527 313-6570 313-6575 317-6652 317-6653 317-6655 317-6656 317-6656 317-6657 317-6665 317-6673 317-6673
        318-6715 318-6715 318-6717 318-6721 318-6721 318-6725 322-6767 322-6771 323-7045 324-7066 324-7104 324-7112 324-7113 327-7153
        334-7771 335-7772
R7       17-0   100-440  100-441  101-445  101-446  101-451  101-452  103-472  104-476  107-533  113-637  114-650  114-662  117-752
        117-757  120-1046 124-1077 126-1133 130-1165 131-1173 131-1175 131-1176 134-1232 136-1234 139-1317 142-1551 145-1616 146-1632
        147-1653 147-1655 147-1657 148-1706 149-1714 150-1752 151-1765 151-1766 151-1770 151-1771 155-2021 161-2102 162-2136 162-2142
        164-2176 165-2222 172-2313 172-2314 179-2435 179-2436 179-2441 179-2445 179-2457 179-2463 183-2521 183-2524 183-2526 183-2530
        183-2535 192-2734 192-2736 192-2744 192-2747 192-2751 192-2754 192-2763 193-2770 193-3003 193-3016 194-3055 194-3064 195-3110
        196-3126 196-3134 196-3142 198-3217 202-3342 202-3343 209-3451 209-3470 210-3514 211-3547 211-3565 212-3611 212-3612 212-3613
        212-3614 212-3616 212-3622 218-3711 218-3712 218-3720 225-4014 225-4022 225-4024 225-4025 225-4026 225-4031 225-4032 225-4035
        225-4045 225-4051 226-4054 226-4055 226-4057 226-4060 226-4065 226-4067 226-4102 229-4154 229-4161 230-4201 230-4203 230-4212
        231-4226 231-4235 231-4236 231-4240 232-4250 233-4311 234-4331 234-4342 234-4360 235-4367 235-4410 235-4417 236-4427 236-4431
        236-4434 236-4445 236-4446 237-4454 237-4465 237-4466 237-4470 237-4505 237-4506 239-4547 239-4571 240-4600 240-4624 243-4677
        243-4701 243-4706 244-4713 244-4716 244-4727 246-4747 246-4771 246-4776 247-5004 248-5021 248-5023 248-5024 248-5027 248-5033
        248-5034 249-5052 249-5063 249-5064 249-5065 249-5070 249-5071 249-5072 249-5073 249-5075 249-5077 250-5102 250-5111 250-5117
        250-5121 251-5127 251-5136 251-5140 251-5143 251-5151 251-5154 253-5175 253-5176 253-5201 259-5323 259-5335 259-5353 259-5356
        259-5357 259-5401 262-5415 263-5423 264-5440 264-5444 266-5452 267-5455 272-5502 281-5562 281-5565 282-5601 283-5625 289-5651
        289-5654 289-5656 289-5657 289-5660 289-5661 289-5662 289-5666 289-5667 290-5673 290-5676 290-5700 290-5703 290-5707 290-5712
        290-5714 290-5720 290-5724 290-5727 290-5731 291-5735 291-5741 291-5744 291-5746 291-5752 291-5756 291-5761 291-5763 291-5767
        291-5773 291-5776 291-6000 291-6004 292-6010 292-6013 292-6015 292-6021 292-6025 292-6030 292-6032 292-6036 293-6041 293-6052
        293-6061 293-6063 293-6101 293-6102 295-6126 295-6127 296-6142 296-6146 297-6170 297-6171 297-6172 297-6173 297-6174 297-6175
        297-6176 297-6177 297-6200 299-6214 299-6223 301-6255 301-6256 301-6260 301-6265 301-6266 302-6303 302-6304 302-6304 302-6305
        302-6306 302-6306 302-6307 303-6331 303-6332 303-6346 303-6347 304-6362 304-6363 304-6371 304-6372 305-6417 305-6420 305-6421
        306-6433 306-6434 307-6451 309-6502 309-6503 309-6506 309-6506 309-6510 309-6512 312-6540 313-6567 313-6576 315-6616 316-6632
        316-6643 317-6654 317-6666 317-6672 317-6672 318-6703 318-6704 318-6705 318-6707 318-6711 318-6722 318-6724 318-6724 318-6731
        319-6753 319-6753 322-6775 322-6776 322-7002 322-7005 322-7010 323-7023 323-7033 323-7054 324-7063 324-7120 326-7150 326-7152
        327-7156 327-7157 333-7742 333-7743 334-7760 334-7761 334-7762 334-7763 334-7764 334-7766 334-7766 334-7767 334-7767 334-7770
        334-7770 335-7773 335-7774 335-7774 335-7775 335-7776 335-7777
RAMPE    22-0
RAMTST   85-146  106-505
RAMZAP   86-157  104-501  104-504  106-506
RBCAI    15-0    77-7    288-5633 289-5644 299-6211 301-6251 303-6353 304-6413 305-6417
RBN2ND   46-0    46-0     46-0     46-0     46-0
RBNCOD   46-0   325-7133
RBNMSK   46-0   180-2475 189-2654 192-2741 325-7137
RBNPRM   46-0    46-0     46-0
RBNSTR   46-0
RBNXOR   46-0    46-0     46-0     46-0
RCC      17-0
RCI      40-0
RCLR     73-0    73-0     83-112   84-117   84-123   84-127   160-2070
RCMD     16-0
RCSRWO   47-0    47-0
RCSRWI   47-0    47-0     47-0     47-0    132-1206 134-1226 146-1640 147-1661 151-1755 151-1756 151-1762 151-1772 151-1775
RCT.EN   46-0   324-7073
```

```
RCT.MT   46-0    324-7072
RCT.PR   46-0    324-7065 324-7072 324-7072
RCT.SE   46-0    324-7072 324-7072
RCTBLK   51-0     51-0    322-6766 324-7101 324-7111 324-7127
RCTOFF   50-0     51-0    324-7064 324-7103
RCY      17-0
RD       24-0
RD.DIA  122-1060 136-1243 289-5634 289-5640
RDCMD    86-163   87-177   87-206   87-211   88-220   91-253   94-313  115-714  116-744  119-1014 119-1020 120-1032 126-1122 126-1126
        129-1150 129-1155 130-1160 156-2030 156-2037 156-2047
RDFST    15-0     94-320  108-600  115-704  118-1002 290-5673 290-5700 290-5707 290-5714 290-5724 290-5731 291-5741 291-5746 291-5756
        291-5763 291-5773 291-6000 292-6010 292-6015 292-6025 292-6032 293-6041 294-6105 295-6127 296-6154 299-6223
RDNXT    15-0     94-321  108-601  110-627  115-706  118-1003 290-5674 290-5701 290-5710 290-5715 290-5725 290-5732 291-5742 291-5747
        291-5757 291-5784 291-5774 291-6001 292-6011 292-6016 292-6026 292-6033 293-6042 293-6047 293-6050 293-6051 293-6053 293-6054
        293-6067 293-6070 293-6071 293-6072 293-6073 293-6074 294-6106 295-6131 295-6132 296-6155 299-6224
RDPF     16-0
READ     40-0
READCD   69-0     69-0     69-0    194-3036 194-3047 324-7116
RECALB   69-0
RECC     16-0
RECCT    16-0
RELES    84-136   85-145   85-152   89-232
REPSTA   51-0     51-0
RETCNT   60-0    246-4754 246-4764
REVEND   49-0     49-0     49-0
REVSTR   48-0     48-0
RGDATH   51-0     93-311  115-707  116-733  158-2057
RGDATL   51-0     93-305   93-307   94-317  115-705  116-727  116-731  158-2053 158-2055
RIB      17-0
RLL      17-0    138-1304 138-1305 138-1311 138-1313 140-1516 140-1517 142-1541 142-1542 143-1564 145-1606 145-1613 156-2045 160-2067
        164-2175 173-2341 177-2424 196-3137 207-3415 207-3416 220-3730 221-3737 221-3740 221-3762 222-3763 230-4172 230-4175 230-4177
        230-4177 230-4200 230-4200 231-4227 231-4227 231-4234 231-4234 233-4305 240-4627 245-4736 245-4740 245-4740 262-5422 266-5453
        267-5461 269-5473 269-5473 269-5474 270-5476 270-5476 273-5516 273-5517 273-5521 289-5652 301-6257 333-7732 333-7736 334-7765
RNGBSH   49-0     49-0    133-1216 152-1777
RNGBSL   49-0     49-0    132-1212 134-1227 152-2000
RPC      17-0
RPE      14-0
RPOK     12-0     13-0
RRDY     14-0     14-0
RRM      16-0
RRSGEN   16-0
RS.BER   97-350  107-535  114-666  115-717  116-747
RSE      16-0
RSPCOF   47-0     47-0    146-1631 150-1743 151-1762
RSPLEN   47-0     47-0     47-0     47-0    131-1167 142-1537 146-1627 147-1660 149-1734 149-1736 151-1755 151-1772
RSPO     48-0    147-1662
RSPPOF   47-0     47-0    150-1743 151-1775
RSPPTR   47-0     47-0    151-1756
RT1      17-0
RTCS     18-0     18-0     18-0
RTCS05   24-0
RTCS06   24-0
RTCS13   24-0
RTCS14   24-0
```



```
RTDS     18-0     18-0     18-0
RTDS06   25-0
RTOEVE   43-0    119-1016
RUNN     69-0
RUPF     15-0     15-0     77-5    149-1735 161-2111 162-2150 164-2167 196-3133 210-3540 234-4355 258-5313 263-5423 333-7727
RVACTV   60-0    232-4260 232-4261 326-7143
RVCBNH   49-0     49-0    324-7062 324-7062 326-7147 326-7147
RVCBNL   48-0     49-0    324-7062 326-7147
RVCBUF   51-0     51-0    322-6770 322-7000 323-7022
RVCEBF   51-0     51-0    327-7154
RVCFLG   51-0     51-0    228-4137 322-7020 326-7144
RVCIND   50-0     50-0
RVCRTY   50-0     50-0    322-6764 323-7041 323-7050
RVCSAV   48-0     48-0
RVCSDI   51-0     51-0    228-4140 232-4263
RVCT     59-0    223-3767 232-4256 232-4262 239-4563 245-4733
RVCTA    49-0     49-0    324-7114 326-7151
RVCVEC   51-0     51-0
RVECDP   51-0     51-0    323-7057 324-7107 324-7124
RVECUP   51-0     51-0    228-4142 228-4145 322-7016 323-7055 324-7105 324-7122
RVWRIT   60-0    231-4224 231-4224 232-4261 233-4316 233-4316 322-7013
RWM      16-0
RWRDY    25-0
S.ADRH   54-0    193-3010 251-5135 251-5146 251-5156
S.ADRL   54-0     54-0    193-3010 193-3010 251-5141 251-5147
S.AXAB  152-2002 153-2004 221-3743 278-5540
S.BITO  140-1530 203-3351 227-4110 227-4112 230-4171 233-4300 234-4342 240-4624 241-4632 282-5603
S.BUFR  108-574  110-620  117-756  117-764  118-774  118-1006 122-1056 122-1062 124-1077 124-1100 125-1104 136-1241 136-1245 137-1253
        137-1256
S.CL11  266-5451 282-5612
S.CLRB  142-1540 163-2154 163-2155 163-2156 165-2231 173-2342 182-2503 182-2504 182-2511 182-2512 188-2624 188-2632 188-2633 188-2634
        188-2635 192-2762 192-2765 193-2772 193-3024 193-3025 196-3135 197-3153 197-3202 203-3353 203-3371 210-3534 211-3542 211-3603
        212-3611 212-3612 212-3613 212-3636 212-3637 213-3645 213-3652 226-4103 233-4314 236-4440 238-4526 238-4527 240-4625 242-4663
        251-5156 257-5234 257-5235 257-5236 284-5630 311-6530 312-6532 312-6533 312-6534 324-7111 325-7135
S.CYLH   54-0     54-0    162-2127 192-2767 192-2767 193-3012 211-3544 322-7001 322-7001
S.CYLL   54-0     54-0    162-2122 192-2765 192-2767 193-3011 211-3543 322-7001
S.DECO  283-5615
S.DECB  145-1610 211-3604 231-4225 248-5035 283-5614 315-6626
S.FBC1  144-1577 185-2556 185-2566 225-4036 276-5532
S.FMB1  140-1520 275-5530
S.FUNB  204-3373 220-3733 221-3753 267-5456 277-5534
S.GRUP   54-0     54-0    192-2767 192-2767 195-3067 225-4016 242-4663 242-4664 251-5126 252-5173
S.INCB  197-3160 197-3163 268-5463 282-5611
S.LBNH   54-0    183-2523 194-3035 197-3155 232-4272 250-5116 251-5124
S.LBNL   54-0    183-2532 194-3035 194-3035 197-3154 232-4272 232-4272 250-5115 251-5125
S.LDIO  146-1636 149-1724 161-2101 162-2145 162-2150 180-2471 218-3714 225-4007 226-4063 226-4066 226-4070 233-4320 237-4452 237-4473
        238-4525 239-4550 239-4553 240-4577 240-4604 242-4662 248-5014 248-5015 250-5101 256-5230 282-5602 299-6232 315-6622 316-6644
        317-6672 318-6732 322-6774 323-7057 324-7107
S.LDCR  282-5567 282-5567 309-6477 309-6501 309-6507 312-6546 312-6550
S.LDIU  125-1105 152-2000 204-3375 237-4505 249-5071 262-5407 282-5600
S.LDQQ  169-2264 179-2430 187-2611 191-2727 192-2732 200-3236 204-3372 209-3442 210-3524 225-4044 234-4365 236-4434 240-4614 241-4633
        241-4647 243-4677 244-4720 245-4744 249-5065 251-5127 251-5135 251-5150 251-5160 252-5166 259-5320 273-5505 282-5572 318-6722
        318-6724 322-7002
S.LDRO  134-1226 143-1562 145-1614 147-1646 147-1654 149-1715 149-1727 149-1732 150-1740 150-1744 150-1746 151-1756 151-1762 154-2010
```

```
        154-2011 161-2103 162-2112 162-2123 163-2163 174-2366 175-2370 176-2421 179-2432 183-2525 184-2536 188-2622 189-2650 189-2653
        192-2740 193-2775 193-3020 193-3032 194-3041 195-3076 195-3100 196-3125 196-3136 197-3146 197-3154 200-3233 200-3241 201-3254
        201-3304 202-3315 202-3322 202-3332 202-3343 204-3377 210-3506 210-3522 211-3565 211-3574 213-3640 214-3657 214-3661 214-3670
        221-3752 224-4001 225-4016 226-4064 227-4115 227-4117 229-4154 229-4161 231-4232 232-4243 232-4266 233-4276 233-4323 234-4360
        235-4367 235-4375 235-4420 236-4432 237-4460 239-4572 242-4664 243-4701 243-4706 244-4713 245-4732 246-4765 246-4776 247-5004
        248-5016 248-5027 248-5031 248-5042 249-5044 249-5052 249-5070 249-5077 253-5201 255-5224 257-5254 257-5267 258-5273 258-5275
        259-5327 259-5341 259-5345 259-5361 259-5365 262-5416 263-5424 270-5475 282-5573 307-6452 312-6543 318-6675 318-6676 324-7114
        324-7120 325-7136 326-7140 326-7151 327-7154
S.LDR1  142-1553 145-1622 147-1647 147-1655 147-1664 147-1671 150-1737 151-1755 152-1777 162-2113 162-2116 162-2124 164-2167 167-2252
        169-2266 172-2317 172-2326 173-2331 174-2354 176-2405 176-2413 177-2426 179-2431 184-2537 187-2610 189-2655 195-3103 197-3204
        198-3212 199-3223 200-3237 200-3247 209-3500 210-3512 210-3523 211-3555 211-3573 212-3614 214-3672 225-4033 226-4071 226-4077
        226-4104 227-4124 228-4137 229-4165 234-4332 235-4373 236-4445 238-4514 238-4521 238-4522 238-4530 238-4540 239-4556 244-4714
        244-4716 246-4747 246-4771 248-5022 248-5024 248-5037 249-5047 249-5051 253-5176 255-5225 257-5237 257-5243 257-5245 257-5247
        257-5251 257-5252 257-5253 257-5257 257-5265 258-5274 258-5314 259-5330 259-5343 259-5366 259-5374 259-5376 267-5454 282-5574
        319-6760 322-6762 322-7012 322-7016 322-7020 323-7041 324-7101 324-7124 324-7127 326-7142 326-7144 327-7157
S.LDR2  104-503  146-1641 162-2122 162-2127 162-2131 163-2151 164-2170 165-2225 172-2322 172-2325 179-2452 180-2474 185-2551 193-3013
        194-3052 195-3112 199-3230 209-3457 209-3473 212-3616 225-4010 233-4324 236-4446 238-4533 241-4645 244-4717 246-4755 246-4774
        249-5074 259-5360 264-5432 273-5507 282-5575 315-6601 317-6650 319-6754 323-7023 323-7033
S.LDR3  143-1565 191-2725 211-3546 212-3622 230-4203 234-4331 240-4605 250-5104 250-5115 251-5126 251-5141 266-5450 282-5576 289-5642
        301-6247 307-6451 312-6535 313-6562 317-6647 317-6664 318-6701 322-7010 324-7063
S.LDR5   79-32   138-1276 282-5577 308-6472 313-6564 315-6607 319-6743 319-6753
S.LDR6  163-2160 179-2433 179-2437 179-2442 193-3022 194-3057 194-3062 195-3111 196-3116 197-3155 198-3215 201-3265 210-3536 213-3644
        239-4551 239-4554 250-5107 250-5116 251-5132 251-5152 252-5173 282-5600 313-6571 318-6713 322-6766 323-7053 324-7064
S.LDR7  134-1231 150-1751 151-1764 179-2443 179-2456 179-2461 192-2733 192-2761 193-3011 194-3056 198-3216 201-3266 210-3535 211-3543
        211-3564 226-4053 226-4056 226-4062 229-4151 259-5321 266-5446 282-5601 299-6231 309-6515 315-6604 316-6636 317-6651 318-6702
        318-6730 322-6770 322-7000 323-7022
S.LDUB  142-1556 232-4250 235-4417 236-4427 239-4547 239-4571 243-4703 244-4727 249-5073 253-5175 259-5354 262-5411 264-5436 282-5577
        318-6731
S.LLB0  164-2166 180-2473 192-2760 193-3014 195-3066 195-3113 216-3700 245-4743 282-5604 322-6773 323-7050
S.LLB6  193-3012 211-3544 225-4011 225-4015 225-4021 251-5157 252-5165 282-5606
S.MLB0  201-3270 282-5605
S.MLB6  252-5170 282-5607
S.OPFL   54-0
S.RELC  195-3110 212-3605 212-3623 213-3643 237-4502 238-4536 239-4564 273-5505
S.RELD  273-5510 273-5516
S.RELF  236-4431 273-5514
S.RLBA  273-5506 273-5512
S.RRR1  131-1176 131-1200 211-3576 235-4400 272-5501 272-5504 308-6466 325-7130 331-7442
S.RRR7  131-1174 139-1316 272-5503 318-6712
S.SDI1  274-5526
S.SDIO  274-5524
S.SDI1  124-1072 140-1527 144-1573 145-1603 185-2552 185-2561 274-5522
S.SECI   54-0     54-0
S.SECS   54-0     54-0    192-2767 225-4010 251-5160
S.SLB1  225-4034 227-4125 246-4773 283-5617
S.SSDI  124-1072 130-1165 267-5460
S.ST10  149-1721 199-3231 218-3720 283-5626 312-6542 315-6623
S.ST11  283-5627
S.STIU  122-1051 283-5624
S.STQQ  128-1146 133-1216 180-2467 209-3443 209-3446 210-3530 234-4366 245-4731 246-4770 248-5034 249-5064 250-5111 251-5146 251-5151
        252-5167 259-5337 283-5613 318-6723 318-6725 322-7005
S.STRO  137-1264 140-1526 145-1615 147-1656 149-1716 150-1754 151-1761 161-2111 176-2420 179-2460 182-2507 192-2764 193-3027 193-3033
        196-3130 200-3246 201-3312 202-3324 203-3370 214-3662 214-3663 221-3754 225-4046 225-4047 226-4052 227-4120 239-4570 242-4665
        242-4666 244-4710 249-5043 249-5045 257-5233 257-5271 259-5334 259-5351 259-5363 259-5372 263-5425 283-5616 309-6513 323-7046
```



```
        326-7152 327-7156 327-7162
S.STR1   91-263  137-1263 139-1321 145-1623 147-1651 147-1670 147-1673 155-2015 173-2333 180-2472 192-2763 195-3074 195-3105 200-3240
        200-3251 210-3527 210-3540 211-3554 235-4402 242-4657 248-5033 249-5050 249-5063 257-5242 257-5244 257-5246 257-5250 257-5256
        257-5266 259-5373 259-5375 262-5421 278-5542 283-5620 313-6574 322-6764 322-6775 322-7017 323-7044 324-7102
S.STR2   87-201   93-306  108-562  108-567  116-730  134-1233 158-2054 163-2152 188-2631 194-3045 194-3051 195-3075 198-3214 199-3232
        202-3337 209-3460 241-4653 242-4661 259-5364 264-5435 283-5621 315-6624 322-6776
S.STR3  232-4255 233-4307 251-5125 283-5622 299-6233
S.STR5  133-1215 137-1273 283-5623 313-6573 316-6645
S.STR6  107-530  117-751  233-4310 241-4652 250-5114 251-5124 251-5140 251-5147 252-5174 283-5624 311-6523 311-6524 311-6525 313-6572
S.STR7  160-2071 179-2447 183-2531 183-2533 210-3537 225-4027 225-4030 283-5625 289-5640 301-6245 307-6450 309-6516
S.STUB  142-1554 225-4043 283-5623
S.SWB1  182-2502 182-2513 235-4374 272-5504 322-6763
S.SXAB  259-5336 259-5344 278-5537 278-5541 278-5542
S.SXB1  204-3376 237-4506 249-5072 262-5410 278-5536 278-5536
S.TRAK   54-0     54-0    192-2767 192-2767 195-3100 225-4021 251-5150 252-5166 252-5171
S.TST   149-1725 166-2240 167-2242
S.XORO  151-1773 184-2544 195-3067 212-3624 282-5610
S.ZBUF  137-1255 197-3200 198-3207 200-3235 267-5457 279-5544
SAREG    41-0     81-62    98-367  126-1126 129-1150 129-1155
SAVADR   51-0     51-0
SAVBUF   51-0     51-0
SAVCNT   51-0     51-0
SAVEDC   51-0     51-0
SAVR7    50-0     50-0    307-6450 318-6675 318-6730
SAVUAR   51-0     51-0    128-1146 278-5542 289-5642 299-6233 301-6247
SC.AOL   56-0    209-3472
SC.BAD   56-0
SC.CNT   57-0
SC.DCL   56-0
SC.DDE   57-0
SC.DIS   56-0    176-2410
SC.DSY   56-0    234-4347
SC.DUP   56-0    176-2415 185-2565
SC.EC1   56-0
SC.EC2   56-0
SC.EC3   56-0
SC.EC4   56-0
SC.EC5   56-0
SC.EC6   56-0
SC.EC7   56-0
SC.EC8   56-0
SC.ECC   56-0    211-3572 234-4357 235-4401 243-4670 323-7032
SC.EDC   57-0    212-3621 235-4426 239-4544 240-4610 243-4676 323-7040
SC.FER   56-0    232-4273 235-4425 236-4430 240-4606
SC.HDR   56-0    211-3551 212-3607 234-4353 324-7117
SC.IMR   57-0    258-5312
SC.INV   56-0    211-3552
SC.IOP   56-0
SC.LVO   57-0
SC.N12   56-0    212-3625
SC.NML   56-0
SC.NOM   57-0    232-4253 239-4545
SC.NVL   56-0    176-2406
SC.ODB   57-0    193-3001
```

LSCS FORM=QUAD

```
SC.ODT   57-0    193-3006
SC.OVR   57-0
SC.PAR   57-0    239-4546
SC.POS   56-0
SC.RRD   57-0
SC.RWR   56-0    197-3147 234-4351
SC.SCN   56-0
SC.SDI   56-0
SC.STO   56-0
SC.UNK   56-0    176-2404
SC.WPH   56-0    194-3042
SC.WPS   56-0    194-3042
SCAN     12-0     12-0      12-0      13-0
SCLR     73-0     73-0      76-0    335-7774
SCMD     16-0
SD       18-0
SDI.1    65-0     65-0    142-1543 142-1544 164-2171 202-3326 205-3401 274-5522 274-5523 274-5524 274-5525 274-5526 274-5527
SDI.12   63-0     63-0    180-2473 189-2645 192-2760 225-4011 225-4023 251-5157 322-6773
SDI.1T   60-0     60-0    140-1533 200-3241 233-4310
SDI.2    65-0     65-0    164-2171 202-3326 205-3401
SDI.2T   62-0    193-2772 234-4345 234-4356 234-4365 235-4402 235-4420 236-4432 240-4614 245-4731
SDI.3    65-0     65-0    164-2171 202-3326 205-3401
SDI.4    65-0     65-0    142-1557 142-1560 164-2171 202-3326 205-3401
SDI.AT   62-0     62-0      62-0      62-0      62-0      62-0      62-0    142-1547 164-2171 172-2316 173-2342 202-3326
SDI.BH   61-0     61-0    197-3173 197-3173 240-4623
SDI.BL   61-0     61-0    197-3173 240-4623 240-4623
SDI.CH   60-0     60-0    162-2124 174-2365 194-3056 195-3075 242-4656 242-4661
SDI.CL   60-0     60-0    162-2123 194-3062 195-3074 322-7004
SDI.CP   59-0     60-0
SDI.CW   62-0     63-0    187-2611 201-3254
SDI.DB   60-0     60-0    225-4047 237-4453 327-7162
SDI.DL   61-0
SDI.DP   63-0     63-0
SDI.EO   60-0     60-0    163-2155 196-3135 238-4533
SDI.E1   60-0     60-0    163-2154 203-3347 203-3353 211-3603 233-4314 236-4440 245-4744 246-4767
SDI.EC   61-0    235-4373
SDI.ER   61-0     61-0      61-0    245-4743
SDI.ES   60-0     61-0    161-2077 161-2103 162-2120 162-2131 174-2362 194-3057 195-3076 233-4323 246-4765 322-7012 326-7142
SDI.FC   63-0     63-0    210-3522
SDI.GC   63-0     63-0    179-2437 189-2647 225-4015
SDI.GO   63-0     63-0    179-2461
SDI.GP   60-0     60-0    193-3032 242-4665 246-4774
SDI.H1   63-0     63-0    193-3013 210-3535 266-5451
SDI.H2   63-0     63-0    192-2761 193-3014 194-3052 210-3536
SDI.I1   61-0     61-0    216-3675
SDI.I2   61-0     61-0    216-3676
SDI.I3   61-0     61-0    216-3677
SDI.L1   63-0     63-0    198-3215 212-3632 322-6765
SDI.L2   63-0     63-0    198-3216 212-3633 322-6765 322-6765
SDI.LL   63-0
SDI.LN   52-0    201-3303
SDI.LT   61-0     61-0      61-0
SDI.M1   63-0     63-0    217-3705
SDI.M2   63-0     63-0    217-3706
```



```
SDI.OE   60-0     60-0    226-4103 227-4116 234-4332 238-4526 238-4527
SDI.OM   60-0     60-0    162-2116 163-2160 172-2325 193-3020 193-3022 213-3644 213-3652 229-4155 233-4324 237-4473 241-4633 241-4652
        248-5014 253-5212
SDI.OP   62-0
SDI.PH   63-0     63-0      63-0
SDI.PL   62-0     62-0
SDI.PQ   60-0     60-0    161-2101 172-2317 180-2471 192-2756 194-3037 199-3230 201-3265 208-3440 218-3712 225-4007 226-4070 227-4113
        227-4122 233-4320 238-4525 239-4550 239-4553 240-4577 240-4604 241-4640 241-4653 242-4662 248-5015 250-5101 256-5230 318-6732
        322-6774 322-7007
SDI.RC   61-0    189-2655 211-3573 246-4755 322-6762
SDI.RH   61-0     61-0    203-3356 233-4312 240-4602 240-4602
SDI.RL   61-0     61-0    203-3355 233-4312 233-4312 240-4602
SDI.RO   60-0     60-0    194-3045
SDI.RS   61-0     61-0    189-2650 202-3315
SDI.RT   63-0     63-0    180-2474 189-2653 192-2740 325-7136
SDI.RV   63-0     63-0    189-2652 323-7053
SDI.S1   61-0     62-0    174-2366 176-2405 201-3275 209-3473
SDI.S2   62-0     62-0    201-3276
SDI.S4   62-0    201-3277
SDI.S5   62-0    201-3300
SDI.S6   62-0    201-3301
SDI.S7   62-0    201-3302
SDI.SD   63-0     63-0
SDI.SL   59-0     59-0    194-3051
SDI.SS   60-0     60-0    225-4030 227-4120 229-4165 231-4225 237-4460
SDI.ST   59-0     59-0    124-1073 142-1545 142-1553 172-2322 174-2354 176-2413 181-2477 186-2577 193-2775 195-3106 197-3204 209-3442
        209-3445 209-3460 224-4001 230-4171 231-4232 232-4243 232-4266 238-4530 241-4632 245-4732 270-5475
SDI.SV   60-0     60-0
SDI.SW   60-0     60-0    140-1535 163-2156 193-3027 197-3146 200-3237 203-3371 209-3500 211-3542 211-3546 211-3554 213-3640 213-3645
        232-4255 233-4307 234-4366 238-4535 239-4570 239-4572 323-7046 325-7135 326-7140
SDI.TG   63-0     63-0    179-2433 189-2646 225-4020 252-5165
SDI.TI   61-0     61-0      61-0    216-3700
SDI.TM   60-0     60-0    145-1610
SDI.TO   61-0     61-0      61-0      61-0
SDI.UB   60-0     60-0    201-3266 211-3564 223-3773 225-4046 226-4062 229-4151 231-4237 237-4467 259-5400 273-5505
SDI.UE   61-0     61-0
SDI.UF   63-0     63-0    194-3041 196-3116 210-3512 217-3703 227-4112
SDI.UG   60-0     60-0    193-3033 195-3066 242-4666 246-4773 251-5132
SDI.UN   61-0     61-0    186-2573 200-3247 202-3322 227-4110
SDI.V1   63-0     63-0    202-3320 212-3626 212-3630 212-3634
SDI.V2   63-0     63-0    202-3321 212-3630 212-3635
SDI.XH   60-0     60-0    193-3002 193-3025 210-3534 250-5107 255-5224
SDI.XL   60-0     60-0    193-3002 193-3002 193-3024 210-3531 250-5104 250-5112 255-5225
SDI.XR   61-0
SDIB.L   64-0     64-0      65-0      65-0      65-0      65-0    124-1075 142-1561 144-1577 145-1612 185-2556 185-2566
SDIBEG   65-0     65-0      65-0
SDIRTY   69-0
SDIS     46-0
SDITMO   69-0
SDITO    46-0
SDPF     16-0
SDTIMO   69-0
SEC      25-0
SECC     16-0
```

LSCS FORM=QUAD

```
SECCT   16-0
SECOND 140-1502 143-1567 145-1624 145-1624
SECSZ   46-0     65-0    140-1513 191-2723 193-2777 210-3532 224-4005 230-4205 243-4673 250-5103 251-5142 253-5177 253-5207 323-7036
SECTR   14-0
SEEK    59-0    195-3070 195-3073 195-3104 224-4002 230-4166 237-4471 237-4476 237-4477 237-4501 238-4531 241-4641 241-4642
SEEKL   69-0
SEKPRV  48-0     48-0
SEKSAV  48-0
SENDS2 126-1110 132-1207
SENDS3 126-1112 133-1214
SENDS4 126-1114 138-1277
SEQ01   78-25    82-76
SEQERR  76-3     76-3     76-3     76-4     77-5     77-6     77-10    77-12    78-23   139-1325 329-7400 333-7445
SEQTST  78-17
SEQUEN  46-0    162-2114
SERIAL  22-0
SES     42-0
SETBAR 306-6450 308-6462 319-6761
SETHVR 331-7440 334-7750
SETLIM 128-1137 134-1230 134-1232
SEX2SK  48-0    225-4027 226-4056 226-4064 227-4117
SF      73-0    138-1300
SFERR   70-0
SFT134 135-1234 136-1235 137-1253
SFT20  100-421  100-443
SGRPCD  69-0
SHFID   91-254  120-1031
SHFRES 306-6450 308-6460 308-6463 308-6466
SIMTST   1-0     87-215   98-373  114-667  120-1026 120-1043 126-1121 132-1210 133-1215 133-1222 138-1300 139-1324 142-1541 142-1541
SLAT    59-0    124-1074 196-3131 200-3245 202-3336 223-3766 232-4256 233-4304 233-4306 239-4563 245-4733 247-5012
SM      73-0     87-214
SPURT  229-4147 229-4150
SRM     16-0
SRSGEN  16-0
SSE     16-0
SSSDIA 124-1073 124-1076
SST     42-0
ST.ABO  55-0    163-2165
ST.AVL  55-0    174-2360 176-2401 188-2643 193-2774 210-3504
ST.CMD  55-0    182-2502 182-2513 206-3407 207-3413 222-3764 333-7733
ST.CMP  55-0    232-4274 240-4607 244-4726
ST.CNT  56-0    140-1536 212-3621 235-4426 239-4544 240-4610 243-4676 323-7040
ST.DAT  55-0    211-3551 212-3607 232-4273 234-4347 234-4353 234-4357 235-4401 235-4425 236-4430 240-4606 240-4617 243-4670 323-7032
       324-7117
ST.DRV  56-0    197-3147 234-4351
ST.HST  55-0    193-3001 193-3006 207-3420 221-3747 221-3751 232-4253 232-4254 239-4545 239-4546 258-5312 264-5441
ST.MFE  55-0    211-3572 212-3625
ST.MSK  55-0    176-2377 211-3552 240-4616
ST.OFL  55-0    175-2367 175-2374 176-2400 176-2412 176-2417 188-2637 188-2641 191-2717 193-2771 208-3437 209-3455 209-3465 209-3477
ST.SHF  55-0    182-2502
ST.SUB  55-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0
        56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0     56-0
        56-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0     57-0
ST.SUC  55-0    170-2303 175-2372 176-2377 188-2640 207-3421 209-3472 211-3542 220-3736 264-5442
```



```
ST.WPR  55-0    194-3043
STARSP  69-0
STBBLG 189-2645 189-2646 189-2647 203-3366
STBFLG 189-2652 201-3275 201-3276 201-3277 201-3300 201-3301 201-3302 202-3320 202-3321 202-3330 202-3331 202-3342 203-3355 203-3356
       203-3365 212-3627 212-3630 212-3632 212-3633 212-3634 212-3635 216-3675 216-3676 216-3677 217-3703 217-3705 217-3706 220-3734
       220-3735 221-3760 221-3761
STBFLH 203-3347 203-3360
STCACH  65-0     65-0    144-1572 144-1573
STDALN  46-0    220-3732
STDTBL 219-3723 333-7731 333-7737
STEP    77-11    77-11    77-11    77-15
STEP.1  88-222  129-1147 129-1147 129-1147
STEP.2 131-1175 132-1202
STEP.3 133-1214
STEP.4 137-1274 138-1274
STEP.X 126-1111 126-1113 126-1115 126-1115 126-1115 138-1303
STEP1   35-0     38-0     79-30    80-35    80-44    80-47    87-213
STEP2   35-0    126-1111
STEP3   35-0     79-33   126-1113
STEP4   35-0    126-1130 138-1277
STEST   13-0
STOP    12-0     86-155   93-311   94-315  107-531  108-551  117-754  117-772  118-1000 119-1022 133-1223 148-1701 156-2034 158-2057
       289-5643 299-6212 299-6221 301-6250
STOPCM  43-0
STOPEN  43-0    119-1015
STROLG 189-2651 189-2657 197-3174 202-3316 202-3344 203-3361 203-3364 203-3365 203-3370 205-3405 214-3665 216-3701 217-3707 217-3710
       220-3726 220-3732 221-3756 221-3757 271-5477 331-7444 334-7752 334-7754 334-7756
STR1R2 193-3002 193-3010 194-3035 197-3175 198-3210 198-3213 232-4272 233-4312 240-4623 326-7150
STRTCD  69-0
STS     42-0     86-164  120-1033
SUPF    15-0     15-0     15-0    138-1306 162-2144 165-2205 235-4422 299-6210 333-7730
SUSP    59-0    163-2162 173-2340 175-2373 195-3065 195-3070 195-3073 196-3132 209-3464 211-3562 224-3775 225-4050 229-4160 229-4164
       230-4204 231-4233 232-4267 237-4471 237-4472 237-4474 241-4634 241-4641 245-4734 247-5010
SWM     16-0
SY      48-0     48-0     48-0    308-6474 309-6477 309-6501 309-6515
SY0     48-0    312-6537 316-6633 317-6646
SY8     48-0     48-0
SYNC    50-0     50-0    140-1505
SYNCH   46-0    140-1507
SYNCL   46-0    140-1506
SZBUFA 279-5543 279-5545
T0      83-77    83-101  331-7426
T1      83-102   83-104  331-7427
T2      84-132   84-135  331-7430
T8      85-146   85-151  331-7436
T9      86-153
TEMP    51-0     51-0    224-3776 225-4044 226-4063
TEMP1   48-0     48-0    149-1721 149-1724
TEMP2   48-0     48-0     79-32   133-1215 138-1276
TEMP3   48-0     48-0    107-530  114-643  117-751  122-1051 125-1105 134-1233
TEST    12-0    115-670  120-1027 142-1541
TESTBL 139-1314 331-7426
TMR.BS  50-0     50-0    140-1501 143-1565 143-1570
TMR.MC  50-0     50-0    140-1500 145-1614 145-1623 214-3663
```

```
TOPOL   69-0
TOPRSP  69-0
TSBRPE 103-467  103-471
TSCRPE  77-14   103-467
TSTCNT  17-0     77-7     79-26    80-36    80-41    80-42    80-52    82-75    82-76    83-111   84-121   84-131   85-147   86-162
        86-166   86-167   86-172   86-173   86-174   86-175   88-227  101-455  126-1121 173-2330 238-4512
TSTMOD 169-2263 170-2276 170-2304 174-2351 183-2517 188-2620 191-2700 191-2707 208-3430 214-3653
TSTRTN  86-171   86-174   88-227   88-230   89-232   91-252   91-253   91-264  101-455  107-536  115-670  115-671  116-750  117-752
        117-755  119-1025 120-1027 120-1030 122-1071 286-5633
U.ABRT 165-2216 170-2274
U.ACC  164-2203 190-2670
U.ALOC 223-3774 224-3774
U.ALOH 225-4040 225-4042 226-4061
U.ALOL 225-4041 226-4062
U.ALOZ 226-4072 226-4075 226-4076 226-4100 226-4103
U.ATTA 164-2172 173-2336
U.ATTN 142-1552 172-2313
U.AVAL 165-2211 174-2344
U.AVLO 174-2345 174-2347
U.AVL1 174-2350 174-2354
U.AVLA 174-2356 174-2362
U.AVLB 174-2357 174-2365
U.AVLC 174-2361 175-2371 175-2373
U.BDSM 230-4170 234-4337
U.BFIL 229-4161 231-4240
U.BFLA 229-4157 229-4163 229-4165
U.BFLC 230-4205 322-7014
U.BFLF 230-4210 230-4217
U.BFLH 231-4226 231-4234 231-4235
U.BFLX 231-4232 231-4236 237-4466
U.BFSV 223-3772 227-4106 229-4147
U.BLST 230-4166 236-4450 236-4451 237-4471
U.BMTA 234-4356 235-4373
U.BMTC 234-4343 235-4372 235-4377 235-4403
U.BMTD 235-4406 235-4415
U.BMTE 235-4371 235-4424 236-4431
U.BMTF 236-4433 236-4445
U.BMTI 236-4447 237-4456 237-4460
U.BMTJ 237-4461 237-4465
U.BMTK 234-4347 234-4351 234-4353 234-4362
U.BMTO 239-4545 243-4705 244-4712
U.BMTR 235-4426 238-4512 243-4676
U.BMTS 239-4553 244-4730
U.BMTT 238-4530 238-4534
U.BMTU 238-4532 238-4535
U.BMTV 238-4535 239-4542
U.BMTW 234-4334 241-4643
U.BMTX 234-4331 237-4470
U.BMTY 229-4156 234-4327
U.BMTZ 234-4336 234-4341
U.BSET 230-4205 235-4403 248-5014
U.BST1 248-5026 249-5070 249-5100
U.C2ED 176-2377 188-2641 189-2660 191-2717 213-3642
U.C2EP 170-2303 176-2402 176-2404 176-2416 176-2420 182-2514 188-2637 206-3407 207-3413 207-3420 207-3421 208-3437 209-3455 220-3736
```



```
        221-3747 221-3751 222-3764 264-5445 333-7733
U.C2EZ 176-2417 191-2714
U.CCD  165-2210 170-2270
U.CIN1 144-1574 144-1601
U.CINT 140-1514 144-1572
U.CKEA 253-5214 253-5217
U.CKEB 243-4673 253-5213 253-5220 253-5222 323-7036
U.CKED 212-3617 230-4217 235-4422 253-5175
U.CKTC 224-3777 237-4471 241-4637 255-5224
U.CLGA 232-4275 233-4300
U.CLGB 233-4301 233-4305 233-4314
U.CLGC 233-4313 233-4315
U.CLGD 233-4304 233-4308
U.CLM1 248-5021 248-5023 249-5075 249-5076
U.CLOG 232-4273 236-4443 239-4566
U.CLRM 248-5020 249-5075 249-5075
U.CLRS 163-2157 175-2375 196-3131 200-3245 201-3255 201-3256 202-3336 210-3515 211-3563 211-3602 224-3775 225-4050 229-4152 229-4160
        229-4164 230-4204
U.CLSX 173-2340 175-2376 231-4233
U.CMDA 149-1735 165-2232 170-2275 170-2277 170-2305 170-2307 171-2311 174-2352 182-2506 183-2516 183-2520 188-2621 191-2701 191-2703
        191-2710 191-2712 192-2742 208-3431 208-3433 208-3435 214-3654 214-3656 214-3660
U.CMDB 182-2511 191-2722 191-2731 192-2755
U.CMDC 182-2510 182-2513
U.CMDE 182-2502 193-3021 209-3467
U.CMER 165-2223 165-2231 219-3722 219-3723
U.CMP  196-3137 196-3140 196-3143
U.CMPR 164-2202 190-2662
U.CNVT 179-2430 192-2767 266-5452
U.CPGC 230-4214 235-4412 237-4503 243-4707
U.CPME 251-5154 251-5157
U.CPRM 226-4055 230-4202 237-4464 250-5101
U.CRCT 191-2723 192-2754 198-3215
U.CROS 237-4501 241-4642 241-4655
U.CRSA 242-4660 242-4662
U.CSDI 142-1555 197-3205 223-3766
U.CSLA 201-3253 202-3336
U.CSPT 179-2462 180-2473 194-3044 210-3532
U.CSTA 173-2343 213-3645
U.CUNK 176-2403 181-2477
U.DAP  165-2214 174-2346
U.DATA 222-3763 222-3763 333-7734 333-7735
U.DERC 239-4571 243-4670
U.DERD 235-4425 236-4430 239-4544 239-4555 239-4560 239-4573
U.DERE 239-4574 326-7143
U.DERF 235-4421 236-4436 236-4437 239-4570
U.DERP 232-4251 233-4324 238-4537 239-4573
U.DIAG  83-77
U.DONE 163-2165 174-2360 175-2367 175-2372 175-2374 182-2505 209-3465 213-3641
U.DONX 174-2362 174-2364 197-3203 211-3553 211-3557 213-3640
U.END   83-100   83-103   84-134   85-140   85-150   85-153   85-153   86-153   87-176   87-213   88-215
U.ER   334-7763 334-7764 334-7767 335-7773 335-7776 335-7777 335-8000
U.ERS  165-2204 191-2672
U.EXLP 222-3763
U.EXSP 221-3737 333-7732
```

```
U.FLU   165-2207 170-2272
U.GCSA  183-2526 183-2531
U.GCSB  183-2522 183-2533
U.GCSC  183-2523 183-2532 183-2535
U.GCST  165-2217 183-2515
U.GETP  248-5030 249-5062 258-5273
U.GLBN  232-4271 236-4441 239-4574 240-4577
U.GLBX  197-3151 240-4602
U.GMST  173-2336 175-2375 195-3102 211-3561 213-3646 233-4302 234-4337 237-4475 247-5011 270-5476
U.GPKA  154-2007 154-2012
U.GPKB  154-2006 154-2011
U.GPKT  148-1677 148-1706 154-2005
U.GREF  170-2300 183-2521 184-2536
U.GRFA  184-2542 184-2550
U.GRFB  184-2545 184-2550
U.GSDA  185-2553 185-2560
U.GSDB  185-2554 185-2556
U.GSDC  185-2562 185-2570
U.GSDD  185-2563 185-2566
U.GSDI  185-2551 188-2636 191-2713 208-3436 209-3452
U.GSDN  185-2552 188-2625
U.GSDW  185-2553 185-2562 186-2572
U.GSDX  186-2602 186-2606
U.GSDY  186-2601 186-2603 186-2607
U.GSST  220-3725 333-7731
U.GTAD  150-1747 151-1766 152-1777 155-2016
U.GUCP  175-2373 187-2610 188-2640 191-2715 193-2770 209-3464 210-3521
U.GUS   188-2617
U.GUST  165-2220 188-2617
U.HERR  160-2067 334-7770
U.HTMO  145-1616 145-1622 146-1635 149-1707 229-4153
U.HWVR  202-3335 214-3666 334-7750 334-7750 334-7750
U.IDLB  142-1545 142-1561
U.IDLC  142-1543 143-1562 147-1663
U.IDLD  143-1564 146-1630
U.IDLE  140-1531 140-1533 140-1534 142-1537 143-1571
U.IMEX  170-2301 170-2303 171-2312 183-2534 215-3674 221-3762 333-7736
U.INTA  155-2023 155-2023 156-2030
U.INTI  147-1662 149-1723 155-2013
U.INTR  127-1136 155-2023 156-2026 156-2030
U.INTV  155-2013 155-2015
U.INTX  155-2023 155-2023 156-2030
U.INVA  258-5312 329-7425
U.INVM  258-5276 258-5311
U.IOCM  192-2757 196-3133
U.IOPO  190-2671 191-2673 191-2676
U.IOP1  191-2704 191-2711
U.IOPA  192-2737 192-2744 192-2746
U.IOPB  191-2724 192-2735 192-2743 192-2756
U.IOPC  191-2677 191-2706 193-2770 196-3143
U.IOPE  194-3061 194-3063 194-3064 195-3065 195-3073 322-7004
U.IOPF  195-3070 195-3072 195-3100
U.IOPG  193-2777 195-3110
U.IOPI  196-3122 196-3125
```



```
U.IOPJ  196-3131 238-4541 239-4567
U.IOPK  196-3115 196-3144
U.IOPL  196-3121 196-3124 196-3127 196-3144 197-3146
U.IOPM  193-3030 196-3114 197-3176
U.IOPN  197-3150 197-3153
U.IOPP  193-3017 193-3032
U.IOPR  190-2663 190-2665 190-2667 191-2705
U.IOPS  191-2726 191-2732
U.IOPW  193-3004 193-3011
U.IOPX  192-2761 322-6777
U.IOPY  192-2764 210-3541
U.IOPZ  194-3034 211-3545
U.IOSA  193-3031 197-3170 197-3172
U.IOSH  197-3162 197-3164
U.L1R1  227-4131 227-4132 227-4133 227-4136
U.LNKA  199-3223 199-3225
U.LNKB  199-3222 199-3224
U.LNKH  163-2153 193-3026 199-3230 208-3456 233-4326 237-4474
U.LNKP  192-2756 199-3222 202-3340 208-3440 222-3765
U.LOGA  200-3245 201-3263
U.LOGS  140-1536 200-3233 223-3767
U.LRR1  230-4167 234-4335 241-4636
U.MAP1  248-5040 249-5051 259-5401
U.MAP2  248-5036 249-5047
U.MIN1  257-5261 257-5263 257-5265
U.MINT  196-3141 226-4101 257-5232
U.MIOA  204-3373 221-3741
U.MIOB  204-3375 264-5437
U.MIOP  204-3372 206-3410 207-3414
U.MLUN  202-3325 205-3401 217-3702
U.MNRD  165-2227 206-3406
U.MNWR  165-2230 207-3413
U.MODL  271-5477 334-7757
U.MWDA  206-3412 207-3420
U.MWDB  207-3415 207-3421
U.NOP   170-2271 170-2273 170-2304
U.ONLO  208-3423 208-3426
U.ONLA  174-2353 208-3436
U.ONLB  208-3427 209-3441
U.ONLC  209-3475 209-3477 210-3505
U.ONLD  210-3520 212-3606
U.ONLE  210-3517 210-3527
U.ONLF  209-3463 211-3546
U.ONLG  211-3573 212-3610 212-3621
U.ONLH  211-3575 211-3600
U.ONLI  211-3572 212-3607
U.ONLL  209-3502 209-3503 211-3550 211-3551 212-3631
U.ONLN  165-2212 208-3424
U.ONLR  211-3563 234-4330
U.ONLS  209-3471 210-3504
U.ONLX  210-3520 211-3554 212-3625
U.ONLY  211-3552 211-3564
U.ONLZ  209-3472 209-3474 209-3500 210-3504 211-3567
U.OPCD  149-1733 163-2164 163-2166 164-2166 256-5231
```

```
U.OPCE  165-2223
U.OPDM  164-2173  165-2224
U.OPIM  164-2175  165-2215
U.OVLA  241-4643  241-4651
U.OVLB  241-4640  241-4644
U.OVLP  234-4333  241-4637
U.PKTZ  197-3201  198-3206
U.PTEL  249-5046  257-5272  258-5314
U.PTL2  259-5315  259-5317  259-5341
U.PTL3  259-5323  259-5325
U.PTL4  259-5323  259-5326
U.PTL5  259-5340  259-5345
U.QSND  177-2427  202-3340
U.R1R0  237-4504  237-4507  237-4511  253-5202  253-5204  253-5205  254-5223
U.R1R1  258-5277  258-5300  258-5301  258-5302  258-5303  258-5304  258-5310
U.RAMD  142-1541  145-1624
U.RAMX  142-1541  142-1541  145-1625
U.RCSA  150-1745  151-1776
U.RCSB  146-1631  148-1705  150-1746
U.RCSC  150-1745  150-1747
U.RCSR  146-1627  148-1703  150-1737
U.RCVE  149-1720  149-1726  149-1731  149-1735
U.RD    164-2200  190-2684
U.RECV  142-1537  147-1660  148-1675  149-1734  149-1736
U.RPL   165-2205  191-2674
U.RRCA  245-4732  245-4740
U.RRCC  246-4745  246-4754
U.RRCE  246-4762  246-4767
U.RRCX  245-4734  245-4735  245-4741
U.RREC  211-3600  234-4363  239-4542  245-4731
U.RSTR   76-0      76-0      76-2      88-231
U.RSVA  167-2241  167-2241  208-3434
U.RSVB  167-2247  167-2247  171-2310  191-2702
U.RTST  166-2233  166-2236  167-2245  167-2250
U.RV3A  322-6771  323-7055  324-7060
U.RV3B  231-4224  322-7015
U.RV3C  322-6774  325-7137
U.RV3D  322-7006  322-7011  322-7016
U.RV57  323-7022  333-7730
U.RV5A  323-7025  323-7033
U.RV5B  324-7066  324-7100
U.RV5C  324-7070  324-7074  324-7076
U.RV5D  324-7066  324-7101
U.RV5E  324-7103  324-7113
U.RV5F  324-7073  324-7111
U.RV5G  324-7076  324-7126
U.RV5J  323-7026  323-7031
U.RV5R  323-7053
U.RV5T  323-7045  323-7054  327-7161  327-7163
U.RV5W  323-7030  323-7047
U.RV5X  323-7032  323-7041
U.RV5Y  323-7043  323-7046
U.RV5Z  323-7034  323-7040  324-7061
U.RV7A  324-7061  327-7153
```



```
U.RVC1  321-6762
U.RVC3  322-6762  324-7110  333-7726
U.RVC5  323-7022  333-7727
U.RVC7  327-7153
U.RVCK  231-4223  232-4257  233-4315  233-4323
U.RVCT  223-3770  228-4137
U.RVFA  233-4316  326-7144
U.RVFB  324-7117  326-7141
U.RVFE  324-7072  324-7114
U.RVFH  323-7027  323-7047  323-7052  326-7140
U.RVRL  326-7141  326-7147  327-7155
U.RVRT  325-7133  326-7151  327-7153
U.RVUX  322-7020  333-7725
U.SABT  222-3766
U.SADQ  153-2004  153-2005  155-2017
U.SADX  146-1640  149-1712  153-2003
U.SBCN  176-2417  191-2716  197-3172  198-3210
U.SCCH  165-2221  214-3653
U.SCID  202-3334  214-3667  334-7751
U.SEKA  161-2077  162-2146
U.SEKB  161-2105  161-2107
U.SEKC  161-2077  162-2112
U.SEKG  162-2114  162-2147
U.SEKH  161-2100  161-2110  162-2115  162-2150
U.SEKI  161-2103  161-2106
U.SEKO  161-2101  223-3766  241-4654
U.SEND  143-1563  146-1627
U.SETS  163-2162  195-3102  196-3132  241-4641
U.SETX  195-3103  237-4500
U.SGRP  195-3071  195-3073  242-4664
U.SLBN  197-3152  236-4442  239-4575  240-4604
U.SLEN  201-3281  202-3337
U.SNDA  147-1647  147-1665  147-1674
U.SNDB  147-1643  147-1645  147-1664
U.ST1U  179-2446  179-2451  179-2457  180-2472
U.STER  221-3737  333-7733
U.STOF  150-1742  150-1743  150-1753
U.STRT  139-1324  139-1500  140-1500  267-5461
U.STSX  195-3107  234-4340  247-5013
U.STUD  165-2224  219-3721
U.SUCH  165-2213  208-3422
U.SUID  201-3314  216-3675  217-3704
U.SUNI  189-2661  201-3260  212-3631  217-3702
U.TBEC  235-4415  240-4624  243-4667
U.TIMR  143-1567  145-1603
U.TMDA  211-3566  230-4220  233-4320  234-4354  234-4362
U.TMOD  209-3466  209-3476  210-3507  233-4321
U.TMRA  145-1610  145-1612
U.UCSR  147-1661  149-1722  151-1755
U.UERD  142-1542  160-2066
U.UERR   80-40    145-1621  147-1653  149-1714  150-1752  151-1770  155-2021  160-2066  160-2071  164-2176  165-2222  196-3142  226-4102  259-5356
        262-5415  267-5455
U.ULKA  173-2341  209-3447  213-3641  218-3711
U.ULNK  147-1657  218-3713  264-5444
```

LSCS FORM=QUAD

```
U.UNEA  232-4246  232-4255  239-4576
U.UNEB  232-4262  232-4266
U.UNEC  232-4252  232-4253
U.UNED  230-4206  232-4241
U.UNEE  232-4242  232-4246  235-4404
U.UNER  230-4212  230-4216  232-4247  232-4247  235-4410  235-4414  239-4552
U.UNRD  230-4207  230-4211  230-4216
U.UNWR  235-4407  235-4414
U.VAXP  155-2014  155-2014  196-3134  239-4543  322-7013  322-7015
U.VAXR  140-1501  155-2023  155-2023  156-2027
U.VCER  164-2176  219-3721
U.WR    164-2201  190-2666
U.WTSA  240-4627  240-4627  241-4631
U.WTST  232-4256  239-4563  240-4625  241-4634
U.XCMA  224-4004  237-4472
U.XCMP  223-3771  256-5230
U.XFCP  143-1564  261-5402
U.ZPSW  201-3264  203-3371  247-5010
U8K      21-0      84-115    84-124   104-500
UBAR     17-0      86-157   104-501   104-502   106-506   106-512   106-515   106-517   106-520   106-527   108-545   108-556   108-560   108-563
        108-565   108-570   108-572   108-576   108-603   109-606   110-611   110-613   110-615   110-617   110-623   110-625   117-762   117-770
        117-772   118-775   118-1002  118-1004  118-1010  118-1012  122-1057  122-1064  122-1066  124-1073  124-1075  124-1101  125-1106  125-1107
        136-1242  136-1247  136-1251  137-1255  140-1533  140-1535  142-1543  142-1544  142-1547  142-1553  142-1557  142-1561  144-1574  144-1577
        144-1600  145-1610  145-1612  145-1624  145-1624  145-1625  145-1626  145-1626  146-1627  146-1631  147-1647  147-1661  148-1703  148-1705
        149-1711  149-1722  150-1737  150-1737  150-1740  150-1740  150-1742  150-1744  150-1744  150-1746  150-1747  150-1751  151-1755  151-1756
        151-1762  151-1764  151-1772  151-1775  155-2015  161-2077  161-2101  161-2103  162-2116  162-2120  162-2123  162-2124  162-2131  163-2154
        163-2155  163-2156  163-2160  172-2315  172-2317  172-2322  172-2325  173-2342  174-2354  174-2362  174-2365  174-2368  176-2405  176-2413
        179-2447  179-2447  179-2460  179-2460  180-2467  180-2470  180-2471  180-2472  180-2472  181-2477  185-2556  185-2557  185-2568  185-2567
        186-2573  186-2577  187-2611  187-2615  189-2645  189-2648  189-2647  189-2650  189-2652  189-2653  189-2655  192-2740  192-2758  192-2780
        192-2761  192-2766  192-2767  193-2772  193-2775  193-3002  193-3013  193-3014  193-3020  193-3022  193-3024  193-3025  193-3027  193-3032
        193-3033  194-3037  194-3041  194-3045  194-3051  194-3052  194-3056  194-3057  194-3062  195-3066  195-3074  195-3075  195-3076  195-3106
        196-3116  196-3133  196-3135  196-3136  197-3146  197-3153  197-3173  197-3204  198-3215  198-3216  199-3230  200-3237  200-3241  200-3247
        201-3254  201-3265  201-3266  201-3275  201-3276  201-3277  201-3300  201-3301  201-3302  202-3315  202-3320  202-3321  202-3322  203-3347
        203-3353  203-3355  203-3356  203-3371  204-3374  205-3401  208-3440  209-3442  209-3445  209-3460  209-3473  209-3500  210-3512  210-3522
        210-3531  210-3534  210-3535  210-3536  211-3542  211-3546  211-3554  211-3564  211-3573  211-3603  212-3626  212-3631  212-3632  212-3633
        212-3634  212-3635  213-3640  213-3644  213-3645  213-3652  216-3675  216-3676  216-3677  216-3700  217-3703  217-3705  217-3706  218-3711
        220-3734  220-3735  221-3754  221-3755  223-3773  224-4001  225-4007  225-4011  225-4015  225-4020  225-4023  225-4030  225-4046  225-4047
        226-4062  226-4070  226-4077  226-4103  227-4110  227-4112  227-4113  227-4116  227-4120  227-4122  228-4141  229-4151  229-4155  229-4165
        230-4171  230-4207  230-4216  231-4225  231-4232  231-4237  232-4243  232-4255  232-4264  232-4266  233-4307  233-4310  233-4312  233-4314
        233-4320  233-4323  233-4324  234-4332  234-4345  234-4356  234-4365  234-4366  235-4373  235-4402  235-4403  235-4416  235-4420  236-4432
        236-4440  237-4453  237-4460  237-4467  237-4473  238-4525  238-4526  238-4527  238-4530  238-4533  238-4535  239-4550  239-4553  239-4570
        239-4572  240-4577  240-4602  240-4604  240-4614  240-4623  241-4632  241-4633  241-4640  241-4652  241-4653  242-4656  242-4661  242-4662
        242-4665  242-4666  243-4672  245-4731  245-4732  245-4743  245-4744  246-4755  246-4765  246-4767  246-4773  246-4774  248-5014  248-5015
        248-5031  248-5035  248-5037  249-5042  249-5043  249-5044  249-5045  249-5047  249-5051  250-5101  250-5104  250-5107  250-5112  251-5132
        251-5157  252-5165  253-5212  255-5224  255-5225  256-5230  257-5233  257-5234  257-5235  257-5236  257-5240  257-5242  257-5244  257-5246
        257-5250  257-5256  257-5257  257-5265  257-5267  257-5270  258-5273  258-5314  259-5315  259-5320  259-5321  259-5327  259-5330  259-5334
        259-5337  259-5341  259-5343  259-5345  259-5353  259-5354  259-5357  259-5361  259-5364  259-5365  259-5366  259-5372  259-5373  259-5374
        259-5375  259-5400  262-5414  266-5452  267-5457  270-5475  273-5505  274-5522  274-5523  277-5534  277-5535  282-5577  289-5663  289-5670
        290-5673  290-5700  290-5707  290-5714  290-5716  290-5724  290-5731  290-5733  291-5741  291-5746  291-5750  291-5756  291-5763  291-5765
        291-5773  291-6000  291-6002  292-6010  292-6015  292-6017  292-6025  292-6032  292-6034  293-6041  293-6043  293-6044  293-6045  293-6046
        293-6052  293-6055  293-6056  293-6057  293-6063  293-6103  294-6107  294-6107  295-6125  295-6133  295-6133  296-6147  296-6154  296-6156
        296-6156  296-6157  296-6157  296-6160  296-6160  296-6161  296-6161  296-6162  296-6162  296-6163  296-6163  296-6164  296-6164  299-6216
        299-6227  301-6263  301-6273  301-6273  301-6276  301-6276  302-6302  302-6302  302-6312  302-6313  302-6314  302-6314  302-6316  302-6316
        302-6320  302-6320  302-6322  302-6322  302-6324  302-6324  303-6335  303-6337  303-6337  303-6341  303-6341  303-6353  303-6355  303-6355
```



```
        304-6375  304-6377  304-6377  304-6401  304-6401  304-6413  304-6415  304-6415  306-6432  306-6437  306-6437  306-6443  318-6732  322-6762
        322-6765  322-6773  322-6774  322-7004  322-7007  322-7012  323-7046  323-7053  325-7135  325-7136  326-7140  326-7142  327-7162  329-7404
        329-7405  329-7406  329-7407  329-7410  329-7411  329-7412  329-7413  329-7414  329-7415  329-7416  329-7417  329-7420  329-7421  329-7422
        329-7423
UBURST   49-0      49-0     122-1052  136-1235  139-1320
UC2EDA  176-2400  176-2413
UCRD     18-0      18-0      18-0      76-2      76-4      80-57     81-72     83-105    83-113    84-115    84-124    84-126    84-130    89-240
         89-241   102-457   103-473   104-500   107-534   107-535   108-554   108-574   114-645   114-653   114-656   114-665   115-674   115-711
        116-722   116-741   126-1110  126-1112  126-1114  129-1154  138-1301  145-1607  145-1617  145-1621  155-2025  156-2046  299-6241  334-7762
        334-7764  335-7773  335-7776  335-7777
UCRS     18-0      18-0      18-0     102-460   145-1604  146-1633  148-1675  155-2023  156-2042
UCRTST   83-102   102-456   102-457
UCSREN   43-0
UDA1     73-0
UDD      18-0      18-0      18-0
UDDI     17-0      77-10     77-12    126-1115  126-1116  126-1117
UDS      18-0      18-0      18-0
UER      17-0      79-27     79-30     79-31     79-33     80-34     80-35     80-36     80-40     80-43     80-44     80-46     80-47     80-51
         80-52     80-54     81-61     81-65     81-66     81-67     81-74     83-101    83-104    84-135    85-141    85-151    86-165    88-216
         89-237    90-251    91-252   160-2066  160-2073  160-2074  160-2076  331-7426  331-7427  331-7430  331-7431  331-7432  331-7433  331-7434
        331-7435  331-7436  331-7437
UF.576   63-0
UF.CMR   63-0      63-0     196-3122  227-4111
UF.CMW   63-0      63-0     196-3122
UF.MSK   63-0     210-3505
UF.RMV   63-0
UF.RPL   63-0
UF.WPH   63-0     194-3042  194-3042
UF.WPS   63-0     194-3042  194-3042  210-3510
UGUCPC  187-2613  187-2616
UGUSTA  188-2625  188-2627
UGUSTB  188-2626  188-2631
UGUSTC  188-2623  188-2630  188-2632
UHANG    80-55     80-55     80-60
UHELP    80-55     80-55     80-55     88-216
UIOPA1  192-2744  192-2747
UIOPER  193-2771  193-2774  193-3001  193-3006  193-3022  194-3043
UMP      13-0
UMPN     12-0      77-5      77-6
UN.BRC  230-4215  244-4711  289-5634  289-5636  306-6450
UN.BUF   65-0      65-0
UN.BWC  235-4413  286-5633  301-6242  301-6243
UNB.CA  243-4675  243-4701
UNB.CB  244-4720  244-4724
UNB.CC  244-4723  244-4726
UNB.CM  235-4405  243-4667
UNB.CX  236-4427  244-4725
UNB.RD  149-1713  150-1750  207-3417  221-3746  230-4211  243-4704  259-5355  289-5634  289-5635  306-6450
UNB.RS   80-57    232-4247  286-5633  286-5633
UNB.SR  243-4704
UNB.WR  147-1652  151-1767  155-2020  206-3411  235-4407  286-5633  301-6242  301-6242
UNC.RM   63-0
UNC.SS   61-0
UNERR    70-0
```