```
;MICROCODE FOR TRYDENT DISK CONTROLLER        October 3,1975 11:48 AM
;
#ALTODEF.SY;
;
;
;
;  TRIDENT DISK MICROCODE - TASK 05 - SERVICE OF OUTPUT FIFO
;  TRIDENT DISK MICROCODE - TASK 17 - SERVICE OF INPUT FIFO
;
;  PROCESSES A KBLK (IN PAGE 0) WHICH LOOKS LIKE:
;        640      POINTER TO DCB
;        641      LAST DRIVE SELECTED
;        642      LAST TRACK SELECTED
;        643      STATUS AT SECTOR MARK
;
;  PROCESSES A DCB WHICH LOOKS LIKE:
;        DCB      TRACK ADDRESS
;        DCB+1    HEAD(LEFT BYTE),  SECTOR(RIGHT BYTE)
;        DCB+2    DRIVE SELECT
;        DCB+3    POINTER TO NEXT DCB
; [
;        DCB+4    FIRST COMMAND
;        DCB+5    WORD COUNT
;        DCB+6    MEMORY ADDRESS FOR DATA TRANSFER
;        DCB+7    ECC0 - TO BE FILLED IN BY READ TASK
;        DCB+8    ECC1 - TO BE FILLED IN BY READ TASK
;        DCB+9    STATUS AT END OF TRANSFER
;        DCB+10   INTERRUPT MASK
; ]
;
;
;        **SPECIAL FUNCTION 2 DEFINITIONS**
;
;    "FOO←KSTATUS"        READS THE STATUS            F2 = 10
;    "MD←KDTA"    READS A DATA WORD          F2 = 06, BUS = 2
;    "KTAG←FOO"   WRITES A TAG INSTRUCTION   F2 = 12
;    "KDTA←FOO"   WRITES A DATA WORD         F2 = 13
;    "WAIT"       IS IDENTICAL TO 'BLOCK'    F2 = 14 OR 15
;    "CLRFIFO"    RESETS THE INPUT FIFO      F2 = 16
;    "CLRERR"     RESETS THE ERROR FLIP-FLOPS         F2 = 17
;
$KSTATUS $L66010, 66010, 000100 ;   DF2=10 (RHS)
$KDTA    $L26013, 14002, 124100;    DF2=13 (LHS); BS=2  (RHS)
$KTAG    $L26012, 00000, 124000;    DF2=12 (LHS) REQUIRES BUS DEF
$WAIT    $L24014, 00000, 00000;     NF2=14
$CLRFIFO $L24016, 00000, 00000;     NF2=16
$CLRERR  $L24017, 00000, 00000;     NF2=17
;
;
;        **HANDY CONSTANTS**
;
$600     $600;
$601     $601;
$602     $602;
$603     $603;
$4000    $4000;
$10000   $10000;
$177777  $177777;
$177700  $177700;
$170000  $170000;
$66000   $66000;
$-5      $177773;
$TAGMASK $17777;
;
;        **R REGISTERS**
```

```
;
$AC0      $R3;
$REG36    $R14;
$LREG     $R40;
$DCBPTR   $R41;
$FROM     $R42;
$LAST1    $R43;
$KTEMP    $R44;
$DCBPTR/R         $R45;
$FROM/R   $R46;
$LAST1/R  $R47;
;
;
;
;
;         **PREDEFINITIONS FOR THE FIRST 16 INSTRUCTIONS**
;
  !17,20,FIXECC,RANDOM,COMPAIR,WDSKBGN,I4,I5,I6,I7,I10,I11,I12,I13,I14,I15,I16,RDSKB
**GN;
START:    NOP;                        RETURN TO EMULATOR
;
;
;         **PREDEFINITIONS FOR TEST CONDITIONS**
;
!1,2,DCB+3,DONE;
!1,2,SENDTRK,HEADSEL;
!1,2,R/WCOM,KBLK-1;
!1,2,CKSEC,NOWAIT;
!1,2,CKMASK,SECTOR;
!1,2,LOOP1,SECOK;
!1,2,MASK,NOMASK;
!1,2,SCOMM,RECHECK;
!1,2.PASTSEC,SCOM;
!1,2,READ,WRITE?;
!1,2,WRITE,NO-R/W;
!1,2,CONT/W,DONE/W;
!1,2,READNXT,READ1ST;
!1,2,CHECK,DCB+4;
!1,2,END/CK,CNT/DONE;
!1,2,SEND/CK,WDS/DONE;
;
;
WDSKBGN: T←77777;
         L←10 XOR T;                MAKE BOOT VECTOR OF 77767
         RMR←LREG;                           SET BOOT LOCUS VECTOR
         L←0;
         DCBPTR/R←L,:DONE;
;
DONE:    T←603, WAIT;               UPDATE THE STATUS AT 643B
         MAR←40+T;
         L←KSTATUS;
         MD←LREG;
;
         T←600;
         MAR←40+T;
         ,TASK,CLRERR;
         MD←0,:KBLK;
;
KBLK-1:  KTAG←66000;                TURN OFF CONTROL FUNCTIONS
KBLK:    T←600;                     SEE IF THERE IS A COMMAND
         MAR←40+T;
         KTAG←0;                    TURN OFF LAST KTAG COMMAND
         L←MD+1,BUS=0;      CHECK FOR NO COMMAND ADDRESS;NO TASK!
         DCBPTR←L,:DCB+3;   IFSO GOTO DONE
;
```

```
DCB+3:   T←2;
         MAR←DCBPTR+T;                GET CHAIN ADDRESS
         T←600;
         L←MD;
         MAR←40 + T;                  PUT ADDRESS IN KBLK
         ,TASK;
         MD←LREG;
;
DCB+2:   MAR←DCBPTR+1;                GET DRIVE SELECT
         T←7777;
         L←MD AND T;
         KTAG←LREG;
         T←601;
         MAR←40 + T;                  UPDATE KBLK+1
         ,TASK;
         MD←LREG;
;
DCB+1L:  MAR←DCBPTR;                  GET HEAD SELECT
         T←177400;                            LEFT BYTE ONLY
         L←MD AND T;
         REG36←L LCY 8;
         T←40000;
         L←REG36 OR T,TASK;
         KTAG←LREG;
;
DCB:     MAR←DCBPTR-1;                GET TRACK SELECT
         T←7777;
         L←MD AND T;
         T←602;
         MAR←40+T;
         KTEMP←L;
         T←KTEMP;
         L←MD-T;
         L←20000 OR T,SH=0;           TEST FOR SAME TRACK ADDRESS
         ,:SENDTRK;          IFKSO GOTO HEADSEL
;
SENDTRK: KTAG←LREG;
         T←602,WAIT;                  WAIT UNTILL POSITIONING DONE
         MAR←40 + T;                  UPDATE KBLK+2
         MD←KTEMP;
HEADSEL: KTAG←0;             CLEAR COMMAND REGISTER
         T←10;
         T←4 OR T;
         L←66000 OR T;
         KTAG←LREG;                   SEND HEAD SELECT COMMAND
         L←0;                WAIT FOR NEXT SECTOR
         LAST1←L;            'LAST1'=0 MEANS 1ST BLOCK
;
         T←3;
DCB+4:   MAR←L←DCBPTR+T;
         DCBPTR←L;
         L←MD,BUS=0;         TEST FOR ALL ZERO COMMAND WORD
         KTEMP←L,:R/WCOM;    IFSO GOTO KBLK-1
;
R/WCOM:  T←100;
         T←200 OR T;
         L←KTEMP AND T;
         ,SH=0;              TEST FOR NO READ/WRITE COMMAND
         L←ONE,:CKSEC;       IFSO GOTO NOWAIT;
;
NOWAIT:  LAST1←L;
;
CKSEC:   SINK←LAST1,BUS=0;   TEST FOR FIRST BLOCK
         ,:CKMASK;                    IFSO GOTO SECTOR
;
```

```
SECTOR:  NOP;
;
LOOP1:   ,TASK;              WAIT FOR NEXT SECTOR
         NOP;
;
         T←3;
         MAR←DCBPTR-T;               GET SECTOR COUNT
         T ← 17;
         L←MD AND T;
         T←KSTATUS.T;
         L←LREG-T,WAIT;              TURN OFF WAKE UP TILL SECTOR
         T←4,SH=0;                   CHECK FOR NEXT SECTOR OK
         L←170000 OR T,:LOOP1;IFSO GOTO SECOK
;
SECOK:   KTAG←LREG;
;
CKMASK:  T←KTEMP;
CKMASK+1:        L←100 AND T;
         T←7400,SH=0;        TEST FOR NOT A READ COMMAND
         T←177 OR T,:MASK;   IFSO GOTO NOMASK
;
NOMASK:  T←7777;
MASK:    T←KTEMP.T;                  IF A READ THEN TURN OFF WRITE
         L←66000 OR T;
         REG36←L;
;
         SINK←LAST1,BUS=0;   TEST FOR FIRST BLOCK
         ,:SCOMM;            IFSO GOTO RECHECK
;
RECHECK: T←3;
         MAR←DCBPTR-T;               GET SECTOR COUNT
         T ← 17;
         L←MD AND T;
         T←KSTATUS.T;
         L←LREG-T;                   TURN OFF WAKE UP TILL SECTOR
         ,SH=0;              CHECK FOR NEXT SECTOR OK
         ,:PASTSEC;          IFSO GOTO SCOM
;
PASTSEC: ,WAIT,:SECTOR;              RESTORE DCBPTR AND RETURN
;
;
SCOMM:   NOP;
SCOM:    KTAG←REG36,TASK;            SEND MODIFIED COMMAND
         NOP;
;
;
         T←KTEMP;
         L←100 AND T;
         ,SH=0;              CHECK FOR NOT A READ COMMAND;NO TASK!
         ,:READ;             IFSO GOTO WRITE?
;
WRITE?:  L←200 AND T;                T STILL CONTAINS COMMAND
         ,SH=0;   SEE IF THERE IS NOT A WRITE COMMAND
         ,:WRITE;            IFSO GOTO NO-R/W
;
NO-R/W:  L←DCBPTR+1,WAIT;            WAIT TILL COMM IS ISSUED
         DCBPTR←L;
         ,TASK;   SEND AND INVALID TAG
         KTAG←2,:DONE/W+2;
;
WRITE:   MAR←L←DCBPTR+1;
         DCBPTR←L;
         L←T←MD + 1;                 GET WORD COUNT+1
         KDTA←LREG;
         MAR←L←DCBPTR+1;             NOW GET THE MEMORY ADDRESS
```

KTAG ← 100

```
        T←-5 + T + 1;
        KDTA←ONE;                   SEND THE START BIT TO DISK
        L←MD;
        FROM←L ;                    FROM = FIRST ADDRESS
        L←FROM+T,TASK;
        LAST1←L;                    LAST1 = LAST ADDRESS-4
;
;
LOOP/W: MAR←T←FROM;
        L←LAST1-T;
        L←2+T,SH<0;         SEE IF YOU ARE WRITING THE LAST WORDS
        FROM←L, :CONT/W;    IFSO GOTO DONE/W
CONT/W: KDTA←MD;
        KDTA←MD,TASK;
        ,:LOOP/W;
;
DONE/W: KDTA←MD;
        KDTA←MD;
;
DONE/W+2:T←177700;                          PUT STATUS IN KTEMP
        L←KSTATUS AND T;            SUCH THAT GOOD STATUS = 1
DCB+9:  T←4;                        SKIP ECC WORDS
        MAR←DCBPTR+T;               UPDATE STATUS
        L←LREG + 1,TASK;
        MD←LREG;
;
        T←5,:DCB+4;
;
READ:   SINK←DCBPTR/R,BUS=0;        CHECK FOR FIRST READ COMMAND
        L←DCBPTR,:READNXT;          IFSO GOTO READ1ST
;
READ1ST: DCBPTR/R←L;                SET READ DCB POINTER
;
READNXT: MAR←L←DCBPTR+1;            GET WORD COUNT
        DCBPTR←L;
        L←MD,TASK;
        LAST1←L;                    SAVE WORD COUNT
;
        L←LAST1+1;                  ALLOW FOR 2 WORDS OF ECC
        KDTA←LREG;                  SEND WORD COUNT+1
        T←KTEMP;
        L←200 AND T;
        T←5,SH=0;           TEST FOR NO CHECK COMMAND
        ,:CHECK;            IFSO GOTO DCB+4:
;
CHECK:  MAR←DCBPTR+1;               GET MEMORY ADDRESS
        T←LAST1-1;
        L←MD+1;
        FROM←L;                     SAVE ADDRESS-1
        MAR←FROM-1;
        L←FROM+T;
        LAST1←L;
        KDTA←MD,TASK;
        KDTA←MD;
;
LOOP/CK: MAR←L←T←FROM+1;
        FROM←L;
        L←LAST1-T-1;
        ,SH<0;              TEST FOR LAST WORD
        L←MD,BUS=0,:END/CK;IFSO GOTO CNT/DONE
;                           TEST FOR ALL ZEROS DATA WORD
END/CK: ,TASK,:SEND/CK;     IFSO GOTO WDS/DONE
;
SEND/CK: KDTA←LREG,:LOOP/CK;
;
```

```
WDS/DONE:NOP;                          INSTRUCTION AFTER A TASK
CNT/DONE:T←5,:DCB+4;                   MUST JUMP TO ODD ADDRESS
;
;
;
;
;*************NOW START THE READ TASK MICROCODE************
;
;
;        **PREDEFINITIONS FOR TEST CONDITIONS**
;
!1,2,GETCNT,DISCARD;
!1,2,CONT/R,DONE/R;
!1,2,RDSKBGN+1,FINISHED;
!1,2,ECCER,DCB+10/R;
;
;
;
RDSKBGN: MAR←L←DCBPTR/R+1,BUS=0; SEE IF THERE IS NO DCBPTR
RDSKBGN+1:DCBPTR/R←L,:GETCNT; IFSO GOTO DISCARD
GETCNT:  T←MD-1;                       FIRST GET COUNT-1
         MAR←L←DCBPTR/R+1;             NOW GET THE MEMORY POINTER
         DCBPTR/R←L;
         L←MD-1;
         FROM/R←L;
         L←FROM/R + T,TASK;
         LAST1/R←L;
;
;LOOP/R: MAR←L←T←FROM/R+1;
;        FROM/R←L;
;        L←LAST1/R-T;
;        ,TASK,SH<0 ; SEE IF WRITING THE LAST WORD
;        MD←KDTA,:LOOP/R;  IFSO GOTO DCB+6/R
LOOP/R:  MAR←T←FROM/R+1;
         L←LAST1/R-T-1;
         L←ONE+T,SH<0;     SEE IF YOU ARE WRITING THE LAST WORDS
         FROM/R←L, :CONT/R;            IFSO GOTO DONE/R
CONT/R:  MD←KDTA,TASK;
         MD←KDTA,:LOOP/R;

DONE/R:  MD←KDTA;
         MD←KDTA;
;
;
;~~SINCE THIS MIGHT BE THE END OF READING,
;   THE READ-TASK WAKE-UP MAY NOT BE ACTIVE,
;   SO THERE MUST NOT BE A 'TASK' UNTIL ALL PROCESSING IS DONE.
;
;~~THIS IS THE HIGHEST PRIORITY TASK, SO IT WOULDN'T HELP ANY WAY.
;
;
DCB+6/R: T←KSTATUS;
         L←177700 AND T,TASK;
         FROM/R←L;
;
         MAR← DCBPTR/R + 1;            THERE NOW MUST BE 4 WORDS LEFT
         L←FROM/R + 1;
         MD←KDTA;
         MAR← DCBPTR/R + 1;
         FROM/R←L;                     TURNS ON BIT 15
         MD←KDTA;
         MAR←L← DCBPTR/R + 1;          NOW ENTER THE FIRST ECC WORD
         DCBPTR/R←L;
         MD←T←KDTA;                    SAVE ECC0 IN T
;
```

OK

```
DCB+7/R: MAR←L← DCBPTR/R + 1;        NOW ENTER THE SECOND ECC WORD
         DCBPTR/R←L;
         MD←L←KDTA OR T;             MD←ECC1; L←ECC1 % ECC0
         T←10,SH=0;          TEST FOR ZERO ECC CODE
         L←FROM/R OR T,:ECCER; IFSO GOTO DCB+10/R
;
ECCER:   FROM/R←L;
DCB+10/R:          MAR←L←DCBPTR/R+1;          UPDATE STATUS
         DCBPTR/R←L;
         MD←FROM/R;
;
         MAR←L←DCBPTR/R+1;           NOW GET THE NEXT COMM
         DCBPTR/R←L;
         T←100;
         L←MD AND T;
         ,SH=0;              SEE IF THE NEXT COMM IS NOT A READ
         MAR←L←DCBPTR/R+1,:RDSKBGN+1; IFSO GOTO FINISHED
;
FINISHED:L←0;
         DCBPTR/R←L,TASK;
         SINK←MD,:RDSKBGN;
;
DISCARD: L←0,CLRFIFO;                THROW AWAY ALL WORDS
         DCBPTR/R←L;
         ,TASK;
         ,:RDSKBGN;
;
;
;************NOW START THE ECC MICROCODE ROUTINE************
; JUMPRAM(0)
;                 AC0!0 = NUMBER
;                 AC0!1 = REFERANCE
;
$NUM     $R50;
$REF     $R51;
$COUNT   $R53;
;
;        **PREDEFINITIONS FOR TEST CONDITIONS**
;
;MINIMUM EXECUTION TIME = 20*.17 = 3.4 US
;MAXIMUM EXECUTION TIME = (11 + ((10+11)/2)*2046*).17 = 3654 US
;
!1,2,CONTECC,SAVENUM;
!1,2,SHIFT,EXITECC;
!1,2,CONT2,ECCERROR;
!1,2,XORBITS,NOXOR;
;
;
FIXECC:  MAR←AC0+1;                  GET REFERANCE NUMBER
         T←3777;
         L←MD AND T,BUS=0;TEST FOR REF = 0
         REF←L,:CONTECC;   IFSO GOTO SAVENUM(ie EXIT IMMEDIATELY)
;
CONTECC: MAR←AC0;                    GET NUMBER
         L←0;
         AC0←L;
         L←MD,TASK;
SAVENUM: NUM←L;
;
         T←AC0;
         L←4000-T;
         T←REF,SH<0;         TEST FOR OVER 2047 LOOPS
         L←NUM-T,:CONT2;     IFSO GOTO ECCERROR

CONT2:   L←NUM,SH=0;         TEST FOR DONE
```

```
        REG36←L LSH 1,:SHIFT;IFSO GOTO EXITECC
;
SHIFT:  T←4000;
        L←REG36 AND T;
        L←AC0+1,SH=0;       TEST FOR NO END AROUND BIT;
        AC0←L,:XORBITS;     IFSO GOTO NOXOR
;
NOXOR:  L←REG36,TASK,:SAVENUM;
;
XORBITS: T←4+T+1;           MAKE T = 4005
        L←REG36 XOR T,TASK,:SAVENUM;
;
ECCERROR:         NOP;
EXITECC: ,SWMODE;
        ,:START;  RETURNS THE NUMBER OF SHIFTS

;
;
; RANDOM NUMBER GENERATOR INSTRUCTION - ACCESSED THROUGH
; JUMPRAM(2)
;
$CONST  $R50;
$FROMR  $R51;
$TOR    $R52;
;
!1,2,STORE,EXITRAN;
;
;
RANDOM: MAR←AC0;                    GET CONSTANT 13849 - 1
        NOP;
        L←MD-1;
        CONST←L;
;
        MAR←L←AC0+1;                GET COUNT-1
        AC0←L;
        T←MD;
;
        MAR←AC0+1;
        L←T←MD+T;
        TOR←L,L←T;
        FROMR←L;
;
        MAR←FROMR;                  GET INITIAL RANDOM NUMBER
        NOP;
        L←MD;
        AC0←L;                      STORE RANDOM NUMBER IN AC0
;
LOOP:   T←AC0;
        L←377 AND T;
        AC0←L LCY 8;
        L←AC0;                      L ← 2↑8 * R
        AC0←L LSH 1;
        L←AC0;                      L ← 2↑9 * R
        AC0←L LSH 1;
        L←AC0 + T;                  L ← (2↑10 +2↑1) * R
        AC0←L LSH 1;
        T←CONST + T + 1;
        L←AC0 + T;                  L ← (2↑11 +2↑2 + 1) * R + C
        AC0←L LCY 8;
;
        MAR←T←FROMR+1;
        L←TOR-T;
        L←FROMR+1,SH=0;             TEST FOR FROMR+1 = TOR
        FROMR←L,TASK,:STORE;        IFSO GOTO EXITRAN
STORE:  MD←AC0,:LOOP;
```

```
;
EXITRAN: ,:EXITECC;                    RETURNS NEW RANDOM NUMBER;

;
;
;
;
;
;
; TWO BUFFER COMPAIR ROUTINE - ACCESSED THROUGH
; JUMPRAM(2)
; THIS ROUTINE DOES A WORD FOR WORD COMPAIR AND RETURNS THE
; NUMBER OF WORDS NOT EQUAL
;
;
!1,2,NOTEQUAL,EQUAL;
!1,2,COMPLOOP,EXITCOMP;
;
;
COMPAIR: MAR←AC0;                      GET FIRST BUFFER ADDR
         NOP;
         L←MD-1,TASK;
         FROMR←L;
;
         MAR←L←AC0+1;                  GET SECOND BUFFER ADDR
         AC0←L;
         L←MD-1,TASK;
         TOR←L;
;
         MAR←AC0+1;                    GET WORD COUNT
         L←0;                          STORE ERROR COUNT IN AC0
         AC0←L;                        INITIALIZE ERROR TO 0
         L←MD-1,TASK;
         COUNT←L;
;
COMPLOOP:MAR←L←FROMR+1;                GET FIRST BUFFER WORD
         FROMR←L;
         T←MD;
;
         MAR←L←TOR+1;                  GET SECOND BUFFER WORD
         TOR←L;
         L←MD XOR T;
         ,SH=0;                        TEST FOR WORDS EQUAL
         L←AC0+1,:NOTEQUAL;            IFSO GOTO EQUAL
NOTEQUAL:AC0←L;
EQUAL:   L←COUNT-1,BUS=0,TASK;         TEST FOR COUNT = 0
         COUNT←L,:COMPLOOP;            IFSO GOTO EXITCOMP
;
EXITCOMP:,:EXITECC;
;

;
;
;THIS IS THE END
```